```
                        title   '8086 Loader BDOS Interface, Oct, 1981'
                    ;*****************************************************************
                    ;*****************************************************************
                    ;**                                                             **
                    ;**  B a s i c   D i s k   O p e r a t i n g   S y s t e m      **
                    ;**          I n t e r f a c e   M o d u l e                    **
                    ;**                                                             **
                    ;*****************************************************************
                    ;*****************************************************************
                    ;
                    ;       Copyright (c) 1978, 1979, 1980,1981
                    ;       Digital Research
                    ;       Box 579, Pacific Grove
                    ;       California
                    ;
                    ;
                    ;       Jan 20, 1982
                    ;
                        noiflist
                    ;
                    ;*********************************************
                    ;
0000                cpmsegment      equ     0       ; base of CP/M-86
0000                ldroffset       equ     0000    ; offset of CCP
0400                bdosoffset      equ     0400h   ; offset of BDOS-86
1200                biosoffset      equ     1200h   ; offset of BIOS-86
0C00                bdoscodesize    equ     0c00h   ; size of BDOS code
1000                bdosdataseg     equ     bdosoffset+bdoscodesize
                    ;
```

cp/m 86 v.1.1 - vol I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, A 93950
SER #__ C81-162

```
                        eject ! include bdosequl.a86     ; symbol definitions
                        ;*****************************************************************
                        ;                                                               *
                        ;                       BDOS symbols:                           *
                        ;                                                               *
                        ;*****************************************************************
                        ;
                        ;
0022                    dvers   EQU     22h                     ;version 2.2
                        ;
                        ;
FFFF                    true    EQU     0ffffh                  ;constant true
0000                    false   EQU     not true                ;constant false
                        ;
                        ;
FFFF                    loader  equ     true
                        ;
FFFF                    on      EQU     0ffffh
0000                    off     EQU     00000h
0000                    testf   EQU     off
                        ;
                        ;       Special 8086 symbols:
                        ;
0000                    b       equ     byte ptr 0
0000                    w       equ     word ptr 0
                        ;
                        ;       bios value defined at end of module
                        ;
0050                    ssize   EQU     80                      ;96 level stack
                        ;
                        ;
0000                    cseg    cpmsegment
                        ;
                        ;       CCP entry
                        ;
                                org     ldroffset
                        ccpent:
                        ;
                        ;    BDOS equates
0005                    list_file       equ     5               ;printer function
000D                    diskio  equ     13              ;diskio function low value
                        ;
                        ;
                        ;
                        ;       BIOS routines
                        ;
                                org     biosoffset
                        bios:
1200                    bootf   EQU     BIOS+3*0        ;cold boot function
1203                    wbootf  EQU     BIOS+3*1        ;warm boot function
1206                    constf  EQU     BIOS+3*2        ;console status function
1209                    coninf  EQU     BIOS+3*3        ;console input function
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, A 93950
SER. #

```
 84
 85  =  120C              conouf  EQU     BIOS+3*4        ;console output function
 86  =  120F              listf   EQU     BIOS+3*5        ;list output function
 87  =  1212              punchf  EQU     BIOS+3*6        ;punch output function
 88  =  1215              readerf EQU     BIOS+3*7        ;reader input function
 89  =  1218              homef   EQU     BIOS+3*8        ;disk home function
 90  =  121B              seldskf EQU     BIOS+3*9        ;select disk function
 91  =  121E              settrkf EQU     BIOS+3*10       ;set track function
 92  =  1221              setsecf EQU     BIOS+3*11       ;set sector function
 93  =  1224              setdmaf EQU     BIOS+3*12       ;set dma function
 94  =  1227              readf   EQU     BIOS+3*13       ;read disk function
 95  =  122A              writef  EQU     BIOS+3*14       ;write disk function
 96  =  122D              liststf EQU     BIOS+3*15       ;list status function
 97  =  1230              sectran EQU     BIOS+3*16       ;sector translate
 98  =  1233              setdmabf EQU    BIOS+3*17       ;set base for disk I/O
 99  =  1236              getsegtabf EQU  BIOS+3*18       ;get segment table address
100  =  1239              getiobf equ     bios+3*19       ;get I/O mapping byte
101  =  123C              setiobf equ     bios+3*20       ;seg I/O mapping byte
102  =  0015              nbiosfuncs      equ     21      ;number of BIOS functions
103  =                    ;
104  =                    ;       equates for non graphic characters
105  =  0003              ctlc    EQU     03h             ;control c
106  =  0005              ctle    EQU     05h             ;physical eol
107  =  0008              ctlh    EQU     08h             ;backspace
108  =  0010              ctlp    EQU     10h             ;prnt toggle
109  =  0012              ctlr    EQU     12h             ;repeat line
110  =  0013              ctls    EQU     13h             ;stop/start screen
111  =  0015              ctlu    EQU     15h             ;line delete
112  =  0018              ctlx    EQU     18h             ;=ctl-u
113  =  001A              ctlz    EQU     1ah             ;end of file
114  =  007F              rubout  EQU     7fh             ;char delete
115  =  0009              tab     EQU     09h             ;tab char
116  =  000D              cr      EQU     0dh             ;carriage return
117  =  000A              lf      EQU     0ah             ;line feed
118  =  005E              ctl     EQU     5eh             ;up arrow
119  =                    ;
120  =                    ;
121  =                    ;       module addresses
122  =                    ;
123  =                    ;       literal constants
124  =  FFFF              enddir  EQU     0ffffh          ;end of directory
125  =                    ;
126  =                    ;
127  =                    ;       file control block (fcb) constants
128  =  00E5              empty   EQU     0e5h            ;empty directory entry
129  =  007F              lstrec  EQU     127             ;last record# in extent
130  =  0080              recsiz  EQU     128             ;record size
131  =  0020              fcblen  EQU     32              ;file control block size
132  =  0004              dirrec  EQU     recsiz/fcblen   ;directory elts / record
133  =  0002              dskshf  EQU     2               ;log2(dirrec)
134  =  0003              dskmsk  EQU     dirrec-1
135  =  0005              fcbshf  EQU     5               ;log2(fcblen)
136  =                    ;
```

```
137
138  =  000C              extnum  EQU     12                      ;extent number field
139  =  001F              maxext  EQU     31                      ;largest extent number
140  =  000D              ubytes  EQU     13                      ;unfilled bytes field
141  =  000E              modnum  EQU     14                      ;data module number
142  =  000F              maxmod  EQU     15                      ;largest module number
143  =  0080              fwfmsk  EQU     80h                     ;file write flag is high
144  =                                                            ;order modnum
145  =  000F              namlen  EQU     15                      ;name length
146  =  000F              reccnt  EQU     15                      ;record count field
147  =  0010              dskmap  EQU     16                      ;disk map field
148  =  001F              lstfcb  EQU     fcblen-1
149  =  0020              nxtrec  EQU     fcblen
150  =  0021              ranrec  EQU     nxtrec+1                ;random record field (2 bytes)
151  =                    ;
152  =                    ;       reserved file indicators
153  =  0009              rofile  EQU     9                       ;high order of first type char
154  =  000A              invis   EQU     10                      ;invisible file in dir command
155  =                    ;       equ     11                      ;reserved
156  =  0060              zero_readro     equ     060h            ;indicates open r/o user 0
157  =  00FE              not_found       equ     0feh
158  =  0008              f8      equ     byte ptr 8
159  =  0080              interfc_bit     equ     80h             ;high bit for interface flag
160  =  007F              not_interfc_bit equ     07fh            ;mask to zero high bit
161  =                    ;
162  =                    ;       Memory management symbols:
163  =                    ;
164  =  0000              membase         equ     word ptr 0
165  =  0002              memlg           equ     word ptr 2
166  =  0004              memdescrlg      equ     4
167  =                    ;
168  =                    ;       interrupt control indicators
169  =                    ;
170  =  00E0              bdosint equ     224             ;BDOS reserved interrupt number
171  =  0200              interruptbit    equ     200h            ; bit 9 of flag word
172  =                    ;
173  =                    ;
174  =                    ;
175  =                    ;********************** end of BDOS symbols **********************
176                       end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject ! include bdosent.a86    ; entry module
                        ;********************* BDOS entry module **********************
                        ;
                        ;       Perform branching, error handling and special
                        ;       8086 stuff.
                        ;
                        ;
                        ;
                        ;
                                if      not loader
                                endif
                        ;
                        ;
                        ;
0000                    cseg    cpmsegment
                        org     bdosoffset-80h          ;dispatch table moved to offset A80h
                                                        ;for release 1.1
                        ;       dispatch table for functions
                        ;
0380 31053F07F705       functab dw      func0,func1,func2,func3
     4407
0388 121290074907               dw      punchf,tolistf,func6,func7
     6707
0390 6C0772077E07               dw      func8,func9,func10,func11
     8507
000C                    diskf   equ     12                      ;beginning of disk functions
0398 EC03430EDA0D               dw      func12,func13,func14,func15
     640E
03A0 DE0EE40E0F0F               dw      func16,func17,func18,func19
     130F
03A8 140F1D0F1E0F               dw      func20,func21,func22,func23
     1F0F
03B0 200F260F2C0F               dw      func24,func25,func26,func27
     330F
03B8 4C09390F3F0F               dw      func28,func29,func30,func31
     400F
03C0 460F470F480F               dw      func32,func33,func34,func35
     490F
03C8 4A0F4B0F8B07               dw      func36,func37,func38,func39
     8307
03D0 870F                       dw      func40
03D2 830F                       dw      func47
03D4 890F                       dw      func48
03D6 F103                       dw      func49                  ;vers 1.1 function
03D8 32053B054505               dw      func50,func51,func52,func53     ;8086 functions
     4605
03E0 460546054605               dw      func54,func55,func56,func57
     4605
03E8 46053005                   dw      func58,func59
                        ;
0029                    nfuncs80        equ     41      ;number of 8080 functions
002F                    sfunc86         equ     47      ;start number for 8086 functions
0036                    nfuncs          equ     54      ;total number of BDOS functions
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #

```
                        ;
                        ;
                        func12:
=03EC B022                      mov     al,dvers
=03EE E99703    0788            jmp     staret
                        ;
                        ;
                        func49:
=03F1 BBA011                    mov     bx,offset dmaad
=03F4 E94A0B    0F41            jmp     sthlret
                        ;
                        ;
                        ;
                        ;
                        ;
                        org     bdosoffset
                        ;
=0400 000000000000              db      0,0,0,0,0,0
                        ;
                        ;       enter here from the user's program with function number in cl,
                        ;       and information address in dx
                        ;
                        ;
                        ;
                        ;
                        bdose:                          ;arrive here from user progs
=0406 1E                        push    ds              ;save user's DS
=0407 8CC8                      mov     ax,cs
=0409 8ED8                      mov     ds,ax           ;set BDOS DS
=040B 8F069D11                  pop     parametersegment        ;user's DS register
                        ;
=                                                       ;switch stack segment
=040F 80F905                    cmp     cl,list_file
=0412 7405      0419            jz      in_bdos         ;if listf....only one user in BDOS
                        ;
=0414 80F90D                    cmp     cl,diskio
=0417 7205      041E            jb      save_stacks     ;console....i/o...allow despool in BDOS
                        in_bdos:
=0419 C6068111FF                mov     mxdos,true
                        ;
                        ;
                        save_stacks:
=041E 8C169411                  mov     SS_save,ss
=0422 89269611                  mov     SP_save,sp
=0426 803E981101                cmp     stack_flag,01
=042B 7407      0434            jz      bdose0
=042D 8CC8                      mov     ax,cs
=042F 8ED0                      mov     ss,ax
=0431 BC5911                    mov     sp,offset endstack
                        ;
                        bdose0:
=0434 8B1E9611                  mov     bx,SP_save      ;enable interrupt if it was on
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
283
284 =0438 8E069411                mov     ES,SS_save      ;upon entry
285 =043C 268B5F04                mov     bx,ES:4[bx]
286 =0440 81E30002                and     bx,interruptbit
287 =0444 7401          0447      jz      bdose1
288 =0446 FB                      sti
289 =                   bdose1:                           ;pick up parameter segment
290 =0447 8CD8                    mov     ax,ds
291 =                             ;
292 =0449 55                      push    bp              ;save registers
293 =044A 57                      push    di
294 =044B 56                      push    si
295 =                             ;
296 =044C 0E                      push    cs              ;switch to local segment registers
297 =044D 07                      pop     es
298 =044E 8C065D10                mov     retrn_es,es     ;return BDOS data base..most calls
299 =                             ;
300 =                             ;
301 =0452 E81A00        046F      call    oldbdos         ;call 8080 BDOS
302 =                             ;
303 =0455 5E                      pop     si              ;unsave 8086 registers
304 =0456 5F                      pop     di
305 =0457 5D                      pop     bp
306 =0458 8E065D10                mov     es,retrn_es           ;return data base in es   by ks
307 =                             ;
308 =                                                     ;switch back to entry stack
309 =045C FA                      cli
310 =045D 8E169411                mov     ss,SS_save
311 =0461 8B269611                mov     sp,SP_save
312 =0465 C606B11100              mov     mxdos,false     ;flag off on exit..no one in DOS
313 =046A 8E1E9D11                mov     ds,parametersegment
314 =046E CF                      iret                    ; return from interrupt
315 =                             ;
316 =                             ;
317 =                   oldbdos:                                ;8080 bdos entry point
318 =046F 89169911                MOV     info,dx                 ;info=dx
319 =0473 88164810                MOV     linfo,dl                ;linfo = low(info) - don't equ
320 =0477 32C0                    XOR     AL,AL
321 =0479 A39B11                  MOV     aret,ax                 ;return value defaults to 0000
322 =047C A25F10                  mov     parcopfl,al             ;set true if parameter
323 =                                                             ;block copied from user
324 =047F A24D10                  MOV     fcbdsk,AL
325 =0482 A24B10                  MOV     resel,AL                ;fcbdsk,resel=false
326 =0485 BB4C0F                  MOV     BX,offset goback        ;common return point
327 =0488 53                      PUSH    BX                      ;jmp goback equivalent to ret
328 =0489 80F929                  CMP     cl,nfuncs80
329 =048C 7209          0497      jb      branch
330 =048E 80E906                  sub     cl,sfunc86-nfuncs80
331 =0491 80F936                  cmp     cl,nfuncs
332 =0494 7201          0497      jb      branch
333 =0496 C3                      RET
334 =                   ;                                       ;skip if invalid #
335 =0497 8AD9          branch: MOV     bl,cl
```

COPYRIGHT 1981,1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
336
337  =0499 B700                    mov     bh,0
338  =049B 8ACA                    MOV     CL,DL               ;possible output char. to CL
339  =049D 03DB                    add     bx,bx               ;branch
340  =049F 2EFFA78003              jmp     cs:functab[bx]
341  =                     ;
342  =                     ;
343  =                     ;       error subroutines
344  =                     persub:                             ;report permanent error
345  =04A4 BB7610                  mov     BX,offset permsg
346  =04A7 E81900          04C3    CALL    errflg              ;to report the error
347  =04AA 3C03                    CMP     AL,ctlc
348  =04AC 7555            0503    jnz     parret              ;rnz
349  =04AE EB10            04C0    jmps    jmp_f0free
350  =                                                         ;reboot if response is ctlc
351  =                                                         ;and ignore the error
352  =                     ;
353  =                     selsub:                             ;report select error
354  =04B0 BB8110                  mov     BX,offset selmsg
355  =04B3 EB08            04BD    JMPS    waiterr             ;wait console before boot
356  =                     ;
357  =                     rodsub:                             ;report write to R/O disk
358  =04B5 BB8D10                  mov     BX,offset rodmsg
359  =04B8 EB03            04BD    JMPS    waiterr
360  =                     ;
361  =                     rofsub:                             ;report read/only file
362  =04BA BB8010                  mov     BX,offset rofmsg    ;wait for console
363  =                     ;
364  =                     waiterr:
365  =                                                         ;wait for response before boot
366  =04BD E80300          04C3    CALL    errflg
367  =                     jmp_f0free:
368  =04C0 E97000          0533    JMP     f0free
369  =                     ;
370  =                     errflg:
371  =                                                         ;report error to console,
372  =                                                         ;message address in BX
373  =04C3 53                      PUSH    BX
374  =04C4 E86701          062E    CALL    crlf                ;stack mssg address, new line
375  =04C7 A0A411                  MOV     AL,curdsk
376  =04CA 0441                    ADD     AL,'A'
377  =04CC A27210                  MOV     dskerr,AL           ;current disk name
378  =04CF B96610                  mov     CX,offset dskmsg
379  =04D2 E86301          0638    CALL    print               ;the error message
380  =04D5 59                      POP     CX
381  =04D6 E85F01          0638    CALL    print               ;error mssage tail
382  =04D9 E96B00          0547    jmp     conin               ;jump to input character
383  =                     ;
384  =                     parsave:
385  =                             ;copy parameterblock from user segment to BDOS segment
386  =                             ;CL-reg = lenght of parameter block
387  =                     ;
388  =04DC 1E                      push    ds
```

```
 387
 390  =04DD 50                       push    ax
 391  =04DE 51                       push    cx
 392  =04DF C6065F10FF               mov     parcopfl,true
 393  =04E4 880E6010                 mov     parlg,cl
 394  =04E8 32ED                     xor     ch,ch
 395  =04EA 8B369911                 mov     si,info
 396  =04EE 8936A911                 mov     infosave,si
 397  =04F2 BF9110                   mov     di,offset loc_par_area
 398  =04F5 893E9911                 mov     info,di
 399  =04F9 8E1E9D11                 mov     ds,parametersegment
 400  =04FD FC                       cld
 401  =04FE F3A4             rep     movs    al,al
 402  =0500 59                       pop     cx
 403  =0501 58                       pop     ax
 404  =0502 1F                       pop     ds
 405  =0503 C3               parret: ret
 406  =                      ;
 407  =                      parsave33:
 408  =                              ;copy 33 byte length parameterblock
 409  =0504 51                       push    cx
 410  =0505 B121                     mov     cl,33
 411  =0507 EB03         050C        jmps    pscommon
 412  =                      ;
 413  =                      parsave36:
 414  =                              ;copy 36 byte length parameterblock
 415  =0509 51                       push    cx
 416  =050A B124                     mov     cl,36
 417  =                      pscommon:
 418  =050C E8CDFF       04DC        call    parsave
 419  =050F 59                       pop     cx
 420  =0510 C3                       ret
 421  =                      ;
 422  =                      parunsave:
 423  =                              ;copy local parameter block to user segment
 424  =                      ;
 425  =0511 06                       push    es
 426  =0512 50                       push    ax
 427  =0513 51                       push    cx
 428  =0514 8A0E6010                 mov     cl,parlg
 429  =0518 32ED                     xor     ch,ch
 430  =051A 8E069D11                 mov     es,parametersegment
 431  =051E BE9110                   mov     si,offset loc_par_area
 432  =0521 8B3EA911                 mov     di,infosave
 433  =0525 893E9911                 mov     info,di
 434  =0529 FC                       cld
 435  =052A F3A4             rep     movs    al,al
 436  =052C 59                       pop     cx
 437  =052D 58                       pop     ax
 438  =052E 07                       pop     es
 439  =052F C3                       ret
 440  =                      ;
 441  =                      ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
442
443 =                   ;****************** end BDOS entry module ****************
444                     end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject ! include bdosldl.a86     ; new bdos load function
                        ;       Function 59 is not included in loader BDOS
                        func59:
0530 C3                         ret
                        ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
452
453  =                      eject ! include bdosnewl.a86    ; new 8086 BDOS functions
454  =                      ;
455  =                      ;******************** New BDOS functions for 8086 ****************
456  =                      ;
457  =                      func0:
458  =0531 C3                       ret
459  =                      ;
460  =                      ;
461  =                      func50:
462  =0532 C3                       ret
463  =                      ;
464  =                      f0free:
465  =                      restore_stack:
466  =0533 C606981100               mov     stack_flag,0
467  =0538 E9C~0C      1203         jmp     wbootf
468  =                      func51:
469  =                              ;set segment base for disk I/O
470  =                      ;
471  =053B 8916A211                 mov     dmabase,dx
472  =053F 8BCA                     mov     cx,dx
473  =0541 E8EF0C      1233         call    setdmbf
474  =0544 C3                       ret
475  =                      ;
476  =                      ;
477  =                      func52:
478  =0545 C3                       ret
479  =                      ;
480  =                      ;       used by memory management routines for stack management
481  =                      ;
482  =                      ;       BX addresses user memory descriptor block
483  =                      ;
484  =                      ;
485  =                              if      not loader
486  =                              endif
487  =                              if      loader
488  =                      func53:
489  =                      func54:
490  =                      func55:
491  =                      func56:
492  =                      func57:
493  =                      func58:
494  =0546 C3                       ret
495  =                              endif
496  =                      ;
497  =                      ;
498  =                      ;***************** End of new 8086 functions ****************
499                         end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
500
501  =                    eject ! include bdosio.a86      ; byte I/O module
502  =                    ;****************** BDOS byte I/O module ******************
503  =                    ;
504  =                    ;
505  =                    ;       console handlers
506  =                    conin:
507  =                                                            ;read console character to A
508  =0547 BB1E10                 mov     bx,offset kbchar
509  =054A 8A07                   MOV     AL,[bx]
510  =054C C60700                 MOV     b[bx],0
511  =054F 0AC0                   OR      AL,AL
512  =0551 7525          0578     jnz     ret0                    ;rnz
513  =                                                            ;no previous char ready
514  =0553 E93F02        0795     JMP     toconinf                  ;get character externally
515  =                                                            ;ret
516  =                    ;
517  =                    conech:
518  =                                                            ;read character with echo
519  =0556 E8EEFF        0547     CALL    conin
520  =0559 E80A00        0566     CALL    echoc
521  =055C 721A          0578     JB      ret0                    ;rc
522  =                                                            ;echo char before return
523  =055E 50                     PUSH    AX
524  =055F 8AC8                   MOV     CL,AL
525  =0561 E89300        05F7     CALL    tabout
526  =0564 58                     POP     AX
527  =0565 C3                     RET                             ;with character in A
528  =                    ;
529  =                    echoc:
530  =                                                            ;echo character if graphic
531  =                                                            ;cr, lf, tab, or backspace
532  =0566 3C0D                   CMP     AL,cr
533  =0568 740E          0578     jz      ret0                    ;rz
534  =056A 3C0A                   CMP     AL,lf
535  =056C 740A          0578     jz      ret0                    ;rz
536  =056E 3C09                   CMP     AL,tab
537  =0570 7406          0578     jz      ret0                    ;rz
538  =0572 3C08                   CMP     AL,ctlh
539  =0574 7402          0578     jz      ret0                    ;rz
540  =0576 3C20                   CMP     AL,' '
541  =0578 C3            ret0:    RET                             ;carry set if not graphic
542  =                    ;
543  =                    conbrk:                                 ;check for character ready
544  =0579 A01E10                 MOV     AL,kbchar
545  =057C 0AC0                   OR      AL,AL
546  =057E 751E          059E     JNZ     conb1                   ;skip if active kbchar
547  =                                                            ;no active kbchar
548  =                                                            ;check external break
549  =0580 E81702        079A     CALL    toconstf
550  =0583 2401                   AND     AL,1
551  =0585 74F1          0578     jz      ret0                    ;rz
552  =                                                            ;character ready, read it
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
553
554 =0587 E80B02       0795        CALL    toconinf            ;to A
555 =058A 3C13                     CMP     AL,ctls
556 =058C 750D         059B        JNZ     conb0               ;check stop screen function
557 =                                                           ;found ctls, read next char
558 =058E E80402       0795        CALL    toconinf            ;to A
559 =0591 3C03                     CMP     AL,ctlc
560 =0593 7403         0598        jz      ctlc_check
561 =                                                           ;not c reboot,
562 =                                                           ;act as if nothing happened
563 =0595 32C0                     XOR     AL,AL
564 =0597 C3                       RET                         ;with zero in accumulator
565 =                       ;
566 =                       ctlc_check:
567 =                               if      not loader
568 =                               endif
569 =                       ;
570 =                       ctlc_check1:
571 =0598 E998FF       0533        jmp     fOfree              ;proceed as if function with 0 data
572 =                       ;
573 =                       ;
574 =                       conb0:
575 =                                                           ;character in accum, save it
576 =059B A21E10                    MOV     kbchar,AL
577 =                       conb1:
578 =                                                           ;return with true set
579 =059E B001                     MOV     AL,1
580 =05A0 C3                       RET
581 =                       ;
582 =                       conout:
583 =                               ;compute character position/write conso
584 =                                                           ;compcol = true if computing
585 =                                                           ;column position
586 =05A1 A01B10                    MOV     AL,compcol
587 =05A4 0AC0                     OR      AL,AL
588 =05A6 7516         05BE        JNZ     compout
589 =                                                           ;write the character,
590 =                                                           ;then compute column
591 =                                                           ;write console char from CL
592 =05A8 51                       PUSH    CX
593 =05A9 E8CDFF       0579        CALL    conbrk              ;check for screen stop func.
594 =05AC 59                       POP     CX
595 =05AD 51                       PUSH    CX                  ;recall/save character
596 =05AE E8EE01       079F        CALL    toconout            ;externally, to console
597 =05B1 59                       POP     CX
598 =05B2 51                       PUSH    CX                  ;recall/save character
599 =                                                           ;may be copying to list dev.
600 =05B3 803EB61100               cmp     listcp,0
601 =05B8 7403         05BD        JZ      L_11
602 =05BA E8D301       0790        CALL    tolistf
603 =                       L_11:                           ;to printer, if so
604 =05BD 59                       POP     CX                  ;recall the character
605 =                       compout:
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
606
607 =05BE 8AC1                    MOV     AL,CL                   ;recall the character
608 =                                                            ;and compute column position
609 =05C0 BB1D10                  mov     BX,offset column                ;A = char, BX = .column
610 =05C3 3C7F                    CMP     AL,rubout
611 =05C5 7416          05DD      jz      ret1                    ;no column change if nulls
612 =05C7 FE07                    INC     b[bx]                   ;column = column + 1
613 =05C9 3C20                    CMP     AL,' '
614 =05CB 7310          05DD      jae     ret1                    ;rnc
615 =                                                            ;not graphic, reset column
616 =05CD FE0F                    DEC     b[bx]                   ;column = column - 1
617 =05CF 8A07                    MOV     AL,[bx]
618 =05D1 0AC0                    OR      AL,AL
619 =05D3 7408          05DD      jz      ret1                    ;return if at zero
620 =                                                            ;not at zero, may be
621 =                                                            ;bacspace or end of line
622 =05D5 8AC1                    MOV     AL,CL                   ;character back to A
623 =05D7 3C08                    CMP     al,ctlh
624 =05D9 7503          05DE      JNZ     notbacksp
625 =                                                            ;backspace character
626 =05DB FE0F                    DEC     b[bx]                   ;column = column - 1
627 =05DD C3            ret1:     RET
628 =                   notbacksp:
629 =                                                            ;not backspace, eol ?
630 =05DE 3C0A                    CMP     AL,lf
631 =05E0 75FB          05DD      jnz     ret1                    ;return if not
632 =                                                            ;end of line, column = 0
633 =05E2 C60700                  MOV     b[bx],0                 ;column = 0
634 =05E5 C3                      RET
635 =                   ;
636 =                   ctlout:
637 =                             ;send C character with possible preceding up-arrow
638 =05E6 8AC1                    MOV     AL,CL
639 =05EB E87BFF        0566      CALL    echoc                   ;try if not graphic or spec.
640 =05EB 730A          05F7      JAE     tabout                  ;skip if graphic, tab,
641 =                                                            ;cr,lf, or ctlh
642 =                                                            ;send preceding up arrow
643 =05ED 50                      PUSH    AX
644 =05EE B15E                    MOV     CL,ctl
645 =05F0 E8AEFF        05A1      CALL    conout                  ;up arrow
646 =05F3 59                      POP     cx
647 =05F4 80C940                  OR      cl,40h                  ;becomes graphic letter
648 =                                                            ;ready to print
649 =                                                            ;(drop through to tabout)
650 =                   ;
651 =                   tabout:
652 =                                                            ;expand tabs to console
653 =05F7 80F909                  CMP     cl,tab
654 =05FA 75A5          05A1      jnz     conout                  ;direct to conout if not
655 =                                                            ;tab encountered, move to
656 =                                                            ;next tab position
657 =                   tab0:
658 =05FC B120                    MOV     CL,' '
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
659
660  =05FE E8A0FF      05A1        CALL    conout              ;another blank
661  =0601 A01D10                  MOV     AL,column
662  =0604 2407                    AND     AL,111b             ;column mod 8 = 0 ?
663  =0606 75F4        05FC        JNZ     tab0                ;back for another if not
664  =0608 C3          ret2:       RET
665  =                 ;
666  =                 ;
667  =                 backup:
668  =                                                         ;back-up one screen position
669  =0609 E80500      0611        CALL    pctlh
670  =060C B120                    MOV     CL,' '
671  =060E E88E01      079F        CALL    toconouf
672  =                 ;           (drop through to pctlh)                 ;
673  =                 pctlh:
674  =                             ;send ctlh to console without affecting column count
675  =0611 B108                    MOV     CL,ctlh
676  =0613 E98901      079F        JMP     toconouf
677  =                                                         ;ret
678  =                 ;
679  =                 crlfp:
680  =                             ;print #, cr, lf for ctlx, ctlu, ctlr functions
681  =                             ;then move to strtcol (starting column)
682  =0616 B123                    MOV     CL,'#'
683  =0618 E886FF      05A1        CALL    conout
684  =061B E81000      062E        CALL    crlf
685  =                                                         ;column = 0, move to
686  =                                                         ;position strtcol
687  =                 crlfp0:
688  =061E A01D10                  MOV     AL,column
689  =0621 3A061C10                cmp     al,strtcol
690  =0625 73E1        0608        jae     ret2                ;rnc
691  =0627 B120                    MOV     CL,' '
692  =0629 E875FF      05A1        CALL    conout              ;print blank
693  =062C EBF0        061E        jmps    crlfp0
694  =                 ;
695  =                 crlf:
696  =                             ;carriage return line feed sequence
697  =062E B10D                    MOV     CL,cr
698  =0630 E86EFF      05A1        CALL    conout
699  =0633 B10A                    MOV     CL,lf
700  =0635 E969FF      05A1        JMP     conout
701  =                                                         ;ret
702  =                 ;
703  =                 print:
704  =                             ;print message until M(BC) = '$'
705  =0638 8BF1                    MOV     SI,CX
706  =063A AC                      LODS    al
707  =063B 3C24                    CMP     AL,'$'
708  =063D 74C9        0608        jz      ret2                ;rz
709  =                                                         ;more to print
710  =063F 41                      INC     CX
711  =0640 51                      PUSH    CX
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
712
713  =0641 8AC8                    MOV     CL,AL                   ;char to C
714  =0643 1E                      push    ds                      ;restore data segment
715  =0644 16                      push    ss
716  =0645 1F                      pop     ds
717  =0646 E8AEFF      05F7        CALL    tabout                  ;another character printed
718  =0649 1F                      pop     ds
719  =064A 59                      POP     CX
720  =064B EBEB        0638        jmps    print
721  =                       ;
722  =                       read:
723  =                               ;read to info address (max length, current length, buffer)
724  =064D A01D10                    MOV     AL,column
725  =0650 A21C10                    MOV     strtcol,AL              ;save start for ctl-x, ctl-h
726  =0653 8B1E9711                  MOV     BX,info
727  =0657 268A0F                    MOV     CL,es:[bx]
728  =065A 43                        INC     BX
729  =065B 53                        PUSH    BX
730  =065C B500                      MOV     CH,0
731  =                                                               ;CH = current buffer length,
732  =                                                               ;CL = maximum buffer length,
733  =                                                               ;BX= next to fill - 1
734  =                       readnx:
735  =                               ;read next character, CX, BX active
736  =065E 51                        PUSH    CX
737  =065F 53                        PUSH    BX                      ;blen, cmax, BX saved
738  =                       readn0:
739  =0660 E8E4FE      0547          CALL    conin                   ;next char in AL
740  =                       ;       AND     AL,7fh                  ;mask parity bit
741  =0663 5B                        POP     BX
742  =0664 59                        POP     CX                      ;reactivate counters
743  =0665 3C0D                      CMP     AL,cr
744  =0667 7503        066C          JNZ     L_19
745  =0669 E9C600      0732 eofre:   JMP     readen
746  =                       L_19:
747  =066C 3C0A                      CMP     AL,lf                   ;end of line?
748  =066E 74F9        0669          jz      eofre                   ;jump to end of read if ctlh
749  =0670 3C08                      CMP     AL,ctlh
750  =0672 750E        0682          JNZ     noth                    ;backspace?
751  =                                                               ;do we have any characters
752  =                                                               ;to backspace ?
753  =0674 0AED                      OR      ch,ch
754  =0676 74E6        065E          JZ      readnx
755  =                                                               ;characters remain in buffer,
756  =                                                               ;backup one
757  =0678 FECD                      DEC     CH                      ;remove one character
758  =067A A01D10                    MOV     AL,column
759  =067D A21B10                    MOV     compcol,AL              ;col > 0
760  =                                                               ;compcol > 0 marks repeat as
761  =                                                               ;length compute
762  =0680 EB51        06D3          jmps    linelen                 ;uses same code as repeat
763  =                       noth:
764  =                                                               ;not a backspace
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
765
766 =0682 3C7F                    CMP     AL,rubout
767 =0684 750C          0692     JNZ     notrub          ;rubout char?
768 =                                                    ;rubout if possible
769 =0686 0AED                   OR      ch,ch
770 =0688 74D4          065E     JZ      readnx          ;skip if len=0
771 =                                                    ;buffer has characters,
772 =                                                    ;resend last char
773 =068A 268A07                 MOV     AL,es:[bx]
774 =068D FECD                   DEC     CH
775 =068F 4B                     DEC     BX              ;al=last char
776 =                                                    ;blen=blen-1,
777 =                                                    ;next to fill - 1 decremented
778 =0690 EB7C          070E     JMps     rdech1          ;act like this is an echo
779 =                        ;
780 =                        notrub:
781 =                                                    ;not a rubout char, check eol
782 =0692 3C05                   CMP     AL,ctle
783 =0694 750C          06A2     JNZ     note            ;physical end line?
784 =                                                    ;yes, save active counters
785 =                                                    ;and force eol
786 =0696 51                     PUSH    CX
787 =0697 53                     PUSH    BX
788 =0698 E893FF        062E     CALL    crlf
789 =069B C6061C1000             MOV     strtcol,0       ;start position = 00
790 =                        to_readn0:
791 =06A0 EBBE          0660     jmps    readn0          ;for another character
792 =                        note:
793 =                                                    ;not end of line, list toggle?
794 =06A2 3C10                   CMP     AL,ctlp
795 =06A4 7506          06AC     JNZ     notp            ;skip if not ctlp
796 =                                                    ;list toggle - change parity
797 =06A6 F616B611               not     listcp          ;listcp = not listcp
798 =06AA E8B2          065E     jmps    readnx          ;for another char
799 =                        notp:
800 =                                                    ;not a ctlp, line delete?
801 =06AC 3C18                   CMP     AL,ctlx
802 =06AE 7514          06C4     JNZ     notx
803 =06B0 5B                     POP     BX              ;discard start position
804 =                                                    ;loop while column > strtcol
805 =                        backx:
806 =06B1 A01C10                 MOV     AL,strtcol
807 =06B4 BB1D10                 mov     bx,offset column
808 =06B7 90                     nop
809 =06B8 3A07                   cmp     al,[bx]
810 =06BA 7391          064D     jnb     read
811 =06BC 90                     nop
812 =06BD FE0F                   DEC     b[bx]           ;column = column - 1
813 =06BF E847FF        0609     CALL    backup          ;one position
814 =06C2 EBED          06B1     jmps    backx
815 =                        notx:
816 =                                                    ;not a control x, control u?
817 =                                                    ;not control-X, control-U?
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #

```
                                                                    
=06C4 3C15                  CMP     AL,ctlu                         
=06C6 7507          06CF    JNZ     notu            ;skip if not    
=                                                   ;delete line (ctlu)
=06C8 E84BFF        0616    CALL    crlfp           ;physical eol   
=06CB 5B                    POP     BX              ;discard starting position
=06CC E97EFF        064D    JMp     read            ;to start all over
=                   notu:                                           
=                                                   ;not line delete, repeat line?
=06CF 3C12                  CMP     AL,ctlr                         
=06D1 7535          0708    JNZ     notr                            
=                   linelen:                                        
=                                                   ;repeat line, or compute
=                                                   ;line len (ctlh)
=                                                   ;if compcol > 0 
=06D3 51                    PUSH    CX                              
=06D4 E83FFF        0616    CALL    crlfp           ;save line length
=06D7 59                    POP     CX                              
=06D8 5B                    POP     BX                              
=06D9 53                    PUSH    BX                              
=06DA 51                    PUSH    CX                              
=                                                   ;bcur, cmax active,
=                                                   ;beginning at BX
=                   rep0:                                           
=06DB 0AED                  or      ch,ch                           
=06DD 740F          06EE    JZ      rep1            ;count len to 00
=06DF 43                    INC     BX                              
=06E0 268A0F                MOV     CL,es:[bx]          ;next to print
=06E3 FECD                  DEC     CH                              
=06E5 51                    PUSH    CX                              
=06E6 53                    PUSH    BX              ;count length down
=06E7 E8FCFE        05E6    CALL    ctlout          ;character echoed
=06EA 5B                    POP     BX                              
=06EB 59                    POP     CX              ;recall remaining count
=06EC EBED          06DB    jmps    rep0            ;for the next character
=                   rep1:                                           
=                                                   ;end of repeat, recall lengths
=                                                   ;original BC remains pushed
=06EE 53                    PUSH    BX              ;save next to fill
=06EF A01B10                MOV     AL,compcol                      
=06F2 0AC0                  OR      AL,AL           ;>0 if computing length
=06F4 7410          0706    jz      backsp1                         
=                                                   ;column position computed
=                                                   ;for ctlh       
=06F6 2A061D10              SUB     AL,column       ;diff > 0       
=06FA A21B10                MOV     compcol,AL      ;count down below
=                                                   ;move back compcol-column
=                   backsp:                                         
=                                                   ;move back one more space
=06FD E809FF        0609    CALL    backup          ;one space      
=0700 FE0E1B10              DEC     compcol                         
=0704 75F7          06FD    JNZ     backsp                          
=                   backsp1:                                        
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
871
872 =0706 EB98        06A0        jmps    to_readn0
873 =                       notr:
874 =                                                   ;not a ctlr, place into buffer
875 =                       rdecho:
876 =0708 43                       INC     BX
877 =0709 268807                   MOV     es:[bx],al          ;character filled to mem
878 =070C FEC5                     INC     CH                  ;blen = blen + 1
879 =                       rdech1:
880 =                                                   ;look for a random contr. char
881 =070E 51                       PUSH    CX
882 =070F 53                       PUSH    BX                  ;active values saved
883 =0710 8AC8                     MOV     CL,AL               ;ready to print
884 =0712 E8D1FE    05E6           CALL    ctlout              ;may be up-arrow C
885 =0715 5B                       POP     BX
886 =0716 59                       POP     CX
887 =0717 268A07                   MOV     AL,es:[bx]          ;recall char
888 =071A 3C03                     CMP     AL,ctlc             ;set flags for reboot test
889 =071C 8AC5                     MOV     AL,CH               ;move length to A
890 =071E 750B      072B           JNZ     notc                ;skip if not a control c
891 =0720 3C01                     CMP     AL,1                ;control C, must be length 1
892 =0722 7507      072B           JNZ     L_23
893 =0724 8CD8                     mov     ax,ds               ;restore the ES register
894 =0726 8EC0                     mov     es,ax               ;to base of operating system
895 =0728 E96DFE    0598           JMP     ctlc_check
896 =                       L_23:                           ;reboot if blen = 1
897 =                                                       ;length not 1, ignore rebout
898 =                       notc:
899 =                                                       ;not reboot, end of buffer?
900 =072B 3AC1                     CMP     AL,CL
901 =072D 7303      0732           JAE     L_24
902 =072F E92CFF    065E           JMP     readnx
903 =                       L_24:                           ;go for another if not
904 =                       readen:
905 =                                                       ;end of read , store blen
906 =0732 5B                       POP     BX
907 =0733 26882F                   MOV     es:[bx],ch          ;M(current len) = B
908 =0736 B10D                     MOV     CL,cr
909 =0738 8CD8                     mov     ax,ds               ;restore the ES register
910 =073A 8EC0                     mov     es,ax               ;to base of operating system
911 =073C E962FE    05A1           JMP     conout              ;return carriage
912 =                                                       ;ret
913 =                       func1:
914 =                               ;return console character with echo
915 =                       ;
916 =073F E814FE    0556           CALL    conech
917 =0742 EB44      0788           JMPS staret
918 =                       ;
919 = 05F7                 func2   EQU     tabout
920 =                               ;write console character with tab expansion
921 =                       ;
922 =                       ;
923 =                       func3:
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
                                ;return reader character
                        ;
=0744 E8CE0A    1215            CALL    readerf
=0747 EB3F      0788            JMPS staret
                        ;
                        ;func4: equated to punchf
                                ;write punch character
                        ;
                        ;func5: equated to listf
                                ;write list character
                                ;write to list device
                        ;
                        func6:
                                ;direct console i/o - read if 0ffh
                        ;
=0749 8AC1                      MOV     AL,CL
=074B FEC0                      INC     AL
=074D 7409      0758            JZ      dirinp                  ;0ffh => 00h, means input mode
=074F FEC0                      INC     al
=0751 754C      079F            jnz     toconouf
=0753 E84400    079A            call    toconstf
=0756 EB30      0788            jmps    staret
                        dirinp:
=0758 E83F00    079A            CALL    toconstf                  ;status check
=075B 0AC0                      OR      AL,AL
=075D 7503      0762            JNZ     L_26
=075F E91508    0F77            JMP     retmon
                        L_26:                                   ;skip, return 00 if not ready
                                                                ;character is ready, get it
=0762 E83000    0795            CALL    toconinf                  ;to A
=0765 EB21      0788            JMPS staret
                        ;
                        func7:
                                ;return io byte
                        ;
=0767 E8CF0A    1239            call    getiobf
=076A EB1C      0788            JMPS staret
                        ;
                        func8:
                                ;set i/o byte
                        ;
=076C 8BCA                      mov     cx,dx           ;iobyte in dx
=076E E8CB0A    123C            call    setiobf
=0771 C3                        RET                             ;jmp goback
                        ;
                        func9:
                                ;write line until $ encountered
                        ;
=0772 8BCA                      MOV     CX,DX                   ;CX=string address
=0774 1E                        push    ds                      ;read from user segment
=0775 8E1E9D11                  mov     ds,parametersegment
=0779 E8BCFE    0638            call    print                   ;out to console
```

CP/M 86 V.1.1 - rev I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81-162

```
 977
 978  =077C 1F                    pop     ds
 979  =077D C3                    ret
 980  =                    ;
 981  =                    func10:
 982  =                            ;read a buffered console line
 983  =077E 8E069D11               mov     es,parametersegment
 984  =0782 E9C8FE       064D      jmp     read
 985  =                    ;
 986  =                    ;
 987  =                    func11:
 988  =                            ;check console status
 989  =                    ;
 990  =0785 E8F1FD       0579      CALL    conbrk
 991  =                                                    ;(drop through to sta$ret)
 992  =                    st$ret:
 993  =                                                    ;store the A register to aret
 994  =0788 A27B11                MOV     lret,al
 995  =                    funcret:                        ;
 996  =078B C3                    RET                     ;jmp goback
 997  =                                                    ;(pop stack for non CP/M func)
 998  =                    ;
 999  =                    setlret1:
1000  =                                                    ;set lret = 1
1001  =078C B001                  MOV     AL,1
1002  =078E EBF8         0788      jmps    staret
1003  =                    ;
1004  =                    ;
1005  =                    ;
1006  =                    tolistf:
1007  =0790 BB0F12                mov     bx,offset listf
1008  =0793 EB0D         07A2      jmps    thebios
1009  =                    ;
1010  =                    toconinf:
1011  =0795 BB0912                mov     bx,offset coninf    ;all console functions will
1012  =                                                    ;use common entry to BIOS
1013  =0798 EB08         07A2      jmps    thebios
1014  =                    ;
1015  =                    toconstf:
1016  =079A BB0612                mov     bx,offset constf    ;all console functions use this
1017  =079D EB03         07A2      jmps    thebios         ;function
1018  =                    ;
1019  =                    toconouf:
1020  =079F BB0C12                mov     bx,offset conouf
1021  =                    ;
1022  =                    thebios:
1023  =                            if      not loader
1024  =                            endif
1025  =                            if      loader
1026  =07A2 FFD3                  call    bx
1027  =                            endif
1028  =07A4 C3                    ret
1029  =                    ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1030
1031  =                     ;
1032  =                     ;    actual entry to  BIOS console and list handlers
1033  =                     ;
1034  =07A5 FFD3            bioscl: call    bx
1035  =07A7 CB                      retf
1036  =                     ;
1037  =                     ;
1038  =                     ;      end of Basic I/O System
1039  =                     ;
1040  =                     ;*****************************************************************
1041                        end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1042
1043  =                     eject ! include bdosf1.a86     ; file system part 1
1044  =                     ;
1045  =                     ;************************************************************************
1046  =                     ;************************************************************************
1047  =                     ;**                                                                    **
1048  =                     ;**  B a s i c   D i s k   O p e r a t i n g   S y s t e m             **
1049  =                     ;**                                                                    **
1050  =                     ;************************************************************************
1051  =                     ;*************************************** ******************************
1052  =                     ;
1053  =                     ;
1054  =                     ;       fixed addresses referenced in bios module are
1055  =                     ;       pererr (0009), selerr (000c), roderr (000f)
1056  =                     ;
1057  =                     ;       error message handlers
1058  =                     ;
1059  =                     ;pererror:
1060  =                                                             ;report permanent error
1061  =                     ;       lxi h,pererr  jmp goerr
1062  =                     ;
1063  =                     ;roderror:
1064  =                                                             ;report read/only disk error
1065  =                     ;       lxi h,roderr   jmp goerr
1066  =                     ;
1067  =                     ;roferror:
1068  =                                                             ;report read/only file error
1069  =                     ;       lxi h,rofer     ;jmp goerr
1070  =                     ;
1071  =                     selerror:
1072  =                                                             ;report select error
1073  =07A8 BB0610                  mov     BX,offset seler
1074  =                     ;
1075  =                     ;
1076  =                     goerr:
1077  =                                                             ;BX = .errorhandler,call subr.
1078  =07AB FF27                    JMP     w[bx]                   ;to subroutine
1079  =                     ;
1080  =                     ;
1081  =                     ;
1082  =                     ;       local subroutines for bios interface
1083  =                     ;
1084  =                     move:
1085  =                             ;move data length of length CL from source DX to
1086  =                             ;destination given by BX
1087  =07AD 51                      push    cx
1088  =07AE B500                    mov     ch,0
1089  =07B0 8BF2                    mov     si,dx
1090  =07B2 8BFB                    mov     di,bx
1091  =07B4 FC                      cld
1092  =07B5 F3A4            rep     movs    al,al
1093  =07B7 59                      pop     cx
1094  =07B8 C3                      ret
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1095
1096 =                    ;
1097 =                    ;
1098 =                    selectdisk:
1099 =                            ;select the disk drive given by curdsk, and fill
1100 =                            ;the base addresses curtrka - alloca, then fill
1101 =                            ;the values of the disk parameter block
1102 =                    ;
1103 =07B9 8A0EA411               MOV     cl,curdsk           ;current disk# to CL
1104 =                                                       ;lsb of DL = 0 if not yet
1105 =                                                       ;logged in
1106 =07BD E85B0A       121B      CALL    seldskf             ;BX filled by call
1107 =                                                       ;BX = 0000 if error,
1108 =                                                       ;otherwise disk headers
1109 =07C0 83FB00               cmp     bx,0
1110 =07C3 7450         0815    jz      ret4
1111 =07C5 8B17                 MOV     DX,[bx]
1112 =07C7 83C302               add     bx,2
1113 =07CA 891E2410             MOV     cdrmaxa,BX
1114 =07CE 83C302               add     bx,2
1115 =07D1 891E2610             MOV     curtrka,BX
1116 =07D5 83C302               add     bx,2
1117 =07D8 891E2810             MOV     curreca,BX
1118 =07DC 83C302               add     bx,2
1119 =
1120 =07DF 87DA                 xchg    bx,dx
1121 =07E1 891E4110             MOV     tranv,bx            ;,tran vector
1122 =                    ;
1123 =07E5 8B1E2A10             mov     bx,buffa            ;save dir buff for context
1124 =07E9 891EAD11             mov     cbuffa,bx
1125 =                    ;
1126 =07ED BB2A10               mov     BX,offset buffa     ;DX= source for move, BX=dest
1127 =07F0 B108                 MOV     CL,addlist
1128 =07F2 E8B8FF       07AD    CALL    move                ;addlist filled
1129 =                                                       ;now fill the disk
1130 =                                                       ;parameter block
1131 =07F5 8B162C10             MOV     dx,dpbaddr
1132 =07F9 BB3210               mov     BX,offset sectpt    ;BX is destination
1133 =07FC B10F                 MOV     CL,dpblist
1134 =07FE E8ACFF       07AD    CALL    move                ;data filled
1135 =                                                       ;set single/double map mode
1136 =0801 A03B10               MOV     al,byte ptr maxall+1 ;largest allocation number
1137 =0804 BB4A10               mov     BX,offset single
1138 =0807 C607FF               MOV     b[bx],true              ;assume a=00
1139 =080A 0AC0                 OR      AL,AL
1140 =080C 7403         0811    JZ      retselect
1141 =                                                       ;high order of maxall not
1142 =                                                       ;zero, use double dm
1143 =080E C60700               MOV     b[bx],false
1144 =                    retselect:
1145 =0811 B0FF                 MOV     AL,true
1146 =0813 0AC0                 OR      AL,AL
1147 =0815 C3             ret4: RET                         ;select disk function ok
```

rz
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1148
1149  =                     ;
1150  =                     home:
1151  =                             ;move to home position, then offset to start of dir
1152  =0816 E8FF09    1218         CALL    homef
1153  =                                                     ;first directory pos. selected
1154  =0819 33C0                   XOR     AX,AX           ;constant zero to accumulator
1155  =081B 8B1E2610               MOV     BX,curtrka
1156  =081F 8907                   MOV     [bx],ax                 ;curtrk=0000
1157  =                                                     ;curtrk, currec both set to 0
1158  =0821 C3                     RET
1159  =                     ;
1160  =                     rdbuff:
1161  =                             ;read buffer and check if ok
1162  =0822 8B165210               mov     dx,arecord
1163  =0826 E8FE09    1227         CALL    readf           ;current drive, track,...
1164  =0829 EB07      0832         jmps    diocomp         ;check for i/o errors
1165  =                     ;
1166  =                     wrbuffl9:
1167  =082B 8A0E4510               mov     cl,wflag
1168  =                     wrbuff:
1169  =                             ;write buffer and check condition
1170  =                             ;write type (wrtype) is in register CL
1171  =                             ;wrtype = 0 => normal write operation
1172  =                             ;wrtype = 1 => directory write operation
1173  =                             ;wrtype = 2 => start of new block
1174  =082F E8F809    122A         CALL    writef          ;current drive, track,...
1175  =                     diocomp:                        ;check for disk errors
1176  =0832 0AC0                   OR      AL,AL
1177  =0834 74DF      0815         jz      ret4            ;rz
1178  =0836 BB0410                 mov     BX,offset pererr
1179  =0839 E96FFF    07AB         JMP     goerr
1180  =                     ;
1181  =                     seekdir:
1182  =                             ;seek the record containing the current dir entry
1183  =083C 8B1EA611               MOV     BX,dcnt         ;directory counter to BX
1184  =0840 B102                   MOV     CL,dskshf
1185  =0842 D3EB                   shr     bx,cl           ;value to BX
1186  =0844 891E5210               MOV     arecord,BX
1187  =0848 891E5710               MOV     drec,BX         ;ready for seek
1188  =                                                     ; jmp seek
1189  =                                                     ;ret
1190  =                     ;
1191  =                     seek:
1192  =                             ;seek the track given by arecord (actual record)
1193  =                     ;
1194  =                     ;
1195  =084C A15210                 mov     ax,arecord      ;compute track/sector
1196  =084F BA0000                 mov     dx,0
1197  =0852 F7363210               div     sectpt          ;dx=sector, ax=track
1198  =0856 52                     push    dx
1199  =0857 8BC8                   mov     cx,ax           ;update curtrk
1200  =0859 8B362610               mov     si,curtrka
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, A 93950
SER. #_____

```
1201
1202  =085D 890C                    mov     [si],cx
1203  =085F 030E3F10                add     cx,offsetv              ;set BIOS track
1204  =0863 E8B809      121E        call    settrkf
1205  =0866 59                      pop     cx                      ;set BIOS sector
1206  =0867 8B164110                mov     dx,tranv
1207  =086B E8C209      1230        call    sectran
1208  =086E 8BCB                    mov     cx,bx
1209  =0870 E9AE09      1221        jmp     setsecf
1210  =                             ;ret
1211  =                       ;
1212  =                       ;
1213  =                       ;     utility functions for file access
1214  =                       ;
1215  =                       dmposition:
1216  =                             ;compute disk map position for vrecord to BX
1217  =0873 BB3410                  mov     bx,offset blkshf
1218  =0876 8A0F                    MOV     CL,[bx]                 ;shift count to CL
1219  =0878 A05010                  MOV     AL,lvrecord             ;current virtual record to A
1220  =087B D2E8                    shr     al,cl
1221  =                             ;A = shr(vrecord,blkshf) = vrecord/2**(sect/block)
1222  =087D 8AE8                    MOV     CH,AL                   ;save it for later addition
1223  =087F B107                    MOV     cl,7
1224  =0881 2A0F                    SUB     cl,[bx]
1225  =0883 A04F10                  MOV     AL,extval               ;extent value and extmsk
1226  =                       ;
1227  =                                                             ;blkshf = 3,4,5,6,7
1228  =                                                             ;CL=4,3,2,1,0
1229  =                                                             ;shift is 4,3,2,1,0
1230  =0886 D2E0                    shl     al,cl
1231  =                       ;arrive here with A = shl(ext and extmsk,7-blkshf)
1232  =0888 02C5                    ADD     AL,CH                   ;add the previous
1233  =                                                             ;shr(vrecord,blkshf) value
1234  =                                                             ;AL is one of the following
1235  =                                                             ;values, depending upon alloc
1236  =                                                             ;bks blkshf
1237  =                                                             ;1K   3     v/8 + extval * 16
1238  =                                                             ;2K   4     v/16+ extval * 8
1239  =                                                             ;4K   5     v/32+ extval * 4
1240  =                                                             ;8K   6     v/64+ extval * 2
1241  =                                                             ;16K  7     v/128+extval * 1
1242  =088A C3                      RET                             ;with dm$position in A
1243  =                       ;
1244  =                       getdm:
1245  =                             ;return disk map value from position given by CX
1246  =088B 8B1E9911                MOV     BX,info                 ;base address of file
1247  =                                                             ;file control block
1248  =088F 83C310                  ADD     BX,dskmap               ;bx=.diskmap
1249  =0892 03D9                    ADD     BX,CX                   ;index by asingle byte value
1250  =0894 803E4A1000              cmp     single,0                ;single byte/map entry?
1251  =0899 7405        08A0        JZ      getdmd                  ;get disk map single byte
1252  =089B 8A1F                    MOV     BL,[bx]
1253  =089D B700                    MOV     BH,0
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1254
1255  =089F C3                          RET                             ;with BX=00bb
1256  =                          getdmd:
1257  =08A0 03D9                        ADD     BX,CX                   ;bx=.fcb(dm+1*2)
1258  =                                                                  ;return double precision value
1259  =08A2 8B1F                        MOV     bx,[bx]
1260  =08A4 C3                          RET
1261  =                          ;
1262  =                          index:
1263  =                                 ;compute disk block number from current FCB
1264  =08A5 E8CBFF       0873           CALL    dmposition              ;0...15 in register AL
1265  =08A8 8AC8                        MOV     CL,AL
1266  =08AA B500                        MOV     CH,0
1267  =08AC E8DCFF       088B           CALL    getdm                   ;value to BX
1268  =08AF 891E5210                    MOV     arecord,BX
1269  =^^B3 C3                          RET
1270  =                          ;
1271  =                          alloct:
1272  =                                 ;called following index to see if block allocated
1273  =08B4 8B1E5210                    MOV     BX,arecord
1274  =0888 08DB                        or      bx,bx
1275  =08BA C3                          RET
1276  =                          ;
1277  =                          atran:
1278  =                                 ;compute actual record address, assuming index called
1279  =08BB 8A0E3410                    MOV     cl,blkshf               ;shift count to reg AL
1280  =08BF 8B1E5210                    MOV     BX,arecord
1281  =08C3 D3E3                        shl     bx,cl
1282  =08C5 891E5410                    MOV     ablock,BX               ;save shifted block #
1283  =08C9 A05010                      MOV     AL,lvrecord
1284  =08CC 22063510                    AND     AL,blkmsk               ;masked value in AL
1285  =08D0 0AD8                        OR      bl,al
1286  =08D2 891E5210                    MOV     arecord,BX              ;arecord=BX or
1287  =                                                                  ;(vrecord and blkmsk)
1288  =08D6 C3                          RET
1289  =                          ;
1290  =                          getexta:
1291  =                                 ;get current extent field address to AL
1292  =08D7 8B1E9911                    MOV     BX,info
1293  =08DB 83C30C                      add     bx,extnum               ;bx=.fcb(extnum)
1294  =08DE 8A07                        mov     al,[bx]
1295  =08E0 C3                          RET
1296  =                          ;
1297  =                          gtfcba:
1298  =                                 ;compute reccnt and nxtrec addresses for get/setfcb
1299  =08E1 BA0F00                      MOV     DX,reccnt
1300  =08E4 03169911                    add     dx,info                 ;DX=.fcb(reccnt)
1301  =08E8 8B1100                      MOV     BX,(nxtrec-reccnt)
1302  =08EB 03DA                        ADD     BX,DX                   ;bx=.fcb(nxtrec)
1303  =08ED C3                          RET
1304  =                          ;
1305  =                          getfcb:
1306  =                                 ;set variables from currently addressed FCB
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. Box X 579
PACIFIC GROVE, CA 93950
SER. #

```
1307
1308  =08LE E8F0FF      08E1        CALL    gtfcba              ;addresses in DX, BX
1309  =08F1 8A07                    MOV     AL,[bx]
1310  =08F3 A25010                  MOV     lvrecord,al         ;vrecord=fcb(nxtrec)
1311  =08F6 87DA                    XCHG    BX,DX
1312  =08F8 8A07                    MOV     AL,[bx]
1313  =08FA A24E10                  MOV     rcount,AL           ;rcount=fcb(reccnt)
1314  =08FD E8D7FF      08D7        CALL    getexta             ;BX=.fcb(extnum)
1315  =0900 A03610                  MOV     AL,extmsk           ;extent mask to a
1316  =0903 2207                    AND     AL,[bx]             ;fcb(extnum) and extmsk
1317  =0905 A24F10                  MOV     extval,AL
1318  =0908 C3                      RET
1319  =                       ;
1320  =                       setfcb:
1321  =                             ;place values back into current fcb
1322  =0909 E8D5FF      08E1        CALL    gtfcba              ;addresses to DX, BX
1323  =090C A04710                  MOV     AL,seqio
1324  =090F 3C02                    CMP     AL,02
1325  =0911 7502        0915        JNZ     setfc1
1326  =0913 32C0                    XOR     AL,AL               ;clear carry if seq i/o
1327  =                       setfc1:
1328  =                                                         ;=1 if sequential i/o
1329  =0915 02065010                add     AL,lvrecord
1330  =0919 8807                    MOV     [bx],al             ;fcb(nxtrec)=vrecord+seqio
1331  =091B 87DA                    XCHG    BX,DX
1332  =091D A04E10                  MOV     AL,rcount
1333  =0920 8807                    MOV     [bx],al             ;fcb(reccnt)=rcount
1334  =0922 C3                      RET
1335  =                       ;
1336  =                       ;
1337  =                       cmpecs:
1338  =                             ;compute checksum for current directory buffer
1339  =0923 B98000                  MOV     cx,recsiz           ;size of directory buffer
1340  =0926 8B1E2A10                MOV     BX,buffa            ;current directory buffer
1341  =092A 32C0                    XOR     AL,AL               ;clear checksum value
1342  =                       cmpec0:
1343  =092C 0207                    ADD     AL,[bx]
1344  =092E 43                      INC     BX
1345  =092F E2FB        092C        loop    cmpec0
1346  =0931 C3                      RET                         ;with checksum in A
1347  =                       ;
1348  =                       setcdisk:
1349  =                             ;set a '1' value in curdsk position of CX
1350  =0932 51                      PUSH    CX                  ;save input parameter
1351  =0933 8A0EA411                MOV     cl,curdsk
1352  =0937 BB0100                  MOV     BX,1                ;number to shift
1353  =093A D3E3                    shl     bx,cl               ;BX = mas  to integrate
1354  =093C 59                      POP     CX                  ;original mask
1355  =093D 0BD9                    or      bx,cx               ;BX = mask or rol(1,curdsk)
1356  =093F C3                      RET
1357  =                       ;
1358  =                       nowrite:
1359  =                             ;return true if dir checksum difference occurred
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
1360
1361  =0940 A12010                    MOV     ax,rodsk
1362  =0943 8A0E4411                  MOV     cl,curdsk
1363  =0947 D3EB                      shr     ax,cl
1364  =0949 2401                      AND     AL,1
1365  =094B C3                        RET                             ;non zero if nowrite
1366  =                       ;
1367  =                       setro:
1368  =                                                               ;set current disk to read only
1369  =094C 8B0E2010                  mov     cx, ddsk
1370  =0950 E8DFFF      0932         CALL    setcdisk                ;sets bit to 1
1371  =0953 891E2010                  MOV     rodsk,BX
1372  =                                                               ;high water mark in directory
1373  =                                                               ;goes to max
1374  =0957 8B163910                  MOV     dx,dirmax
1375  =095B 42                        INC     dx
1376  =095C 8B1E2410                  MOV     BX,cdrmaxa              ;BX = .cdrmax
1377  =0960 8917                      MOV     [bx],dx                 ;cdrmax = dirmax
1378  =0962 C3                        RET
1379  =                       ;
1380  =                       ckrodir:
1381  =                               ;check current directory element for read/only status
1382  =0963 E81B00      0981         CALL    getdptra                ;address of element
1383  =                       ;
1384  =                       ckrofile:
1385  =                               ;check current buff(dptr) or fcb(0) for r/o status
1386  =0966 8A0900                    MOV     DX,rofile
1387  =0969 03DA                      ADD     BX,DX                   ;offset to r/o bit
1388  =096B 8A07                      MOV     AL,[bx]
1389  =096D D0D0                      RCL     AL,1
1390  =096F 731B        098C         jae     ret5                    ;rnc
1391  =0971 BB0A10                    mov     BX,offset rofer
1392  =0974 EB08        097E         jmps     jmp_goerr                  ;
1393  =                       ;       jmp roferror ;exit to read only disk message
1394  =                       ;
1395  =                       ;
1396  =                       checkwrite:
1397  =                               ;check for write protected disk
1398  =0976 E8C7FF      0940         CALL    nowrite
1399  =0979 7411        098C         jz      ret5                    ;rz
1400  =097B BB0810                    mov     BX,offset roderr
1401  =                       jmp_goerr:
1402  =097E E92AFE      07AB         JMP     goerr
1403  =                       ;       jmp roderror ;read only disk error
1404  =                       ;
1405  =                       getdptra:
1406  =                               ;compute the address of a director element at
1407  =                               ;positon dptr in the buffer
1408  =0981 8B1E2A10                  MOV     BX,buffa
1409  =0985 A05610                    MOV     AL,dptr
1410  =                       addh:
1411  =                               ;BX = BX + AL
1412  =0988 B400                      mov     ah,0
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #

```
1413
1414  =098A 03D8                    add     bx,ax
1415  =098C C3              ret5:   ret
1416  =                     ;
1417  =                     ;
1418  =                     getmodnum:
1419  =                             ;compute the address of the module number
1420  =                             ;bring module number to accumulator
1421  =                             ;(high order bit is fwf (file write flag)
1422  =098D 8B1E9911                MOV     BX,info
1423  =0991 83C30E                  add     bx,modnum
1424  =0994 8A07                    MOV     AL,[bx]
1425  =0996 C3                      RET                             ;AL=fcb(modnum)
1426  =                     ;
1427  =                     clrmodnum:
1428  =                             ;clear the module number field for user open/make
1429  =0997 E8F3FF      098D       CALL    getmodnum
1430  =099A C60700                  MOV     b[bx],0                 ;fcb(modnum)=0
1431  =099D C3                      RET
1432  =                     ;
1433  =                     setfwf:
1434  =099E E8ECFF      098D       CALL    getmodnum               ;BX=.fcb(modnum),
1435  =                                                            ;AL=fcb(modnum)
1436  =                                                            ;set fwf(file write flag) to 1
1437  =09A1 0C80                    OR      AL,fwfmsk
1438  =09A3 8807                    MOV     [bx],al                 ;fcb(modnum)=fcb(modnum) + 80h
1439  =                                                            ;also returns non zero
1440  =                                                            ;in accumulator
1441  =09A5 C3                      RET
1442  =                     ;
1443  =                     ;
1444  =                     compcdr:
1445  =                             ;return cy if cdrmax > dcnt
1446  =09A6 8B16A611                MOV     dx,dcnt                 ;DX = directory counter
1447  =09AA 8B1E2410                MOV     BX,cdrmaxa              ;BX=.cdrmax
1448  =09AE 3B17                    cmp     dx,[bx]
1449  =                                                            ;condition dcnt - cdrmax
1450  =                                                            ;produces cy if cdrmax>dcnt
1451  =09B0 C3              ret6:   RET
1452  =                     ;
1453  =                     setcdr:
1454  =                             ;if not (cdrmax > dcnt) then cdrmax = dcnt+1
1455  =09B1 E8F2FF      09A6       CALL    compcdr
1456  =09B4 72FA        09B0       jb      ret6                    ;return if cdrmax > dcnt
1457  =                                                            ;otherwise, BX = .cdrmax+1,
1458  =                                                            ;DX = dcnt
1459  =09B6 42                      INC     DX
1460  =09B7 8917                    MOV     [bx],dx
1461  =09B9 C3                      RET
1462  =                     ;
1463  =                     subdh:
1464  =                             ;compute BX = DX - BX
1465  =09BA 52                      push    dx
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1466
1467  =09BB 2BD3                    sub     dx,bx
1468  =09BD 8BDA                    mov     bx,dx
1469  =09BF 5A                      pop     dx
1470  =09C0 C3                      RET
1471  =                       ;
1472  =                       newchecksum:
1473  =                               ;drop through to compute new checksum
1474  =09C1 B1FF                      MOV     CL,true
1475  =                       checksum:
1476  =                               ;compute current checksum record and update the
1477  =                               ;directory element if CL=true, or check for = if not
1478  =                               ;drec < chksiz?
1479  =09C3 8B165710                  MOV     DX,drec
1480  =09C7 8B1E3D10                  MOV     BX,chksiz
1481  =09CB E8ECFF      09BA          CALL    subdh                   ;DX-BX
1482  =09CE 73E0        09B0          jae     ret6                    ;skip checksum if past
1483  =                                                               ;checksum vector size
1484  =                                                               ;drec < chksiz, so continue
1485  =09D0 51                        PUSH    CX                      ;save init flag
1486  =09D1 E84FFF      0923          CALL    cmpecs                  ;check sum value to AL
1487  =09D4 8B1E5710                  MOV     BX,drec                 ;value of drec
1488  =09D8 031E2E10                  ADD     BX,checka               ;bx=.check(drec)
1489  =09DC 59                        POP     CX                      ;recall true or false to CL
1490  =09DD FEC1                      INC     CL                      ;true produces zero flag
1491  =09DF 740D        09EE          JZ      initcs
1492  =                                                               ;not initialzing, compare
1493  =09E1 3A07                      CMP     AL,[bx]                 ;compute$cs=check(drec)?
1494  =09E3 740B        09F0          jz      ret7                    ;no message if ok
1495  =                                                               ;checksum error, are we beyond
1496  =                                                               ;the end of the disk?
1497  =09E5 E8BCFF      09A6          CALL    compcdr
1498  =09E8 7306        09F0          jae     ret7                    ;no message if so
1499  =09EA E85FFF      094C          CALL    setro                   ;read/only disk set
1500  =09ED C3                        RET
1501  =                       initcs:
1502  =                                                               ;initializing the checksum
1503  =09EE 8807                      MOV     [bx],al
1504  =09F0 C3                ret7:   RET
1505  =                       ;
1506  =                       ;
1507  =                       wrdir:
1508  =                               ;write the current directory entry, set checksum
1509  =09F1 E8CDFF      09C1          CALL    newchecksum             ;initialize entry
1510  =09F4 E81900      0A10          CALL    setdir                  ;directory dma
1511  =09F7 B101                      MOV     CL,1                    ;indicates a write directory
1512  =09F9 E833FE      082F          CALL    wrbuff                  ;write the buffer
1513  =09FC EB06        0A04          jmps     setdata                 ;to data dma address
1514  =                                                               ;ret
1515  =                       ;
1516  =                       rddirbuf:
1517  =                               ;read a directory entry into the directory buffer
1518  =09FE E80F00      0A10          CALL    setdir                  ;directory dma
```

```
1519
1520  =0A01 E81EFE      0822        CALL    rdbuff                  ;directory record loaded
1521  =                                                             ; jmp setdata
1522  =                                                             ; ret
1523  =                     ;
1524  =                     setdata:
1525  =                             ;set data dma address
1526  =0A04 8B0EA211                mov     cx,dmabase
1527  =0A08 E82808      1233        call    setdmbf         ;set disk I/O base
1528  =0A0B BBA011                  mov     BX,offset dmaad
1529  =0A0E EB08        0A18        jmps    setdma                  ;to complete the call
1530  =                     ;
1531  =                     setdir:
1532  =                             ;set directory dma address
1533  =0A10 8CD9                    mov     cx,ds
1534  =0A12 E81E08      1233        call    setdmbf                 ;set BIOS disk I/O base
1535  =0A15 B82A10                  mov     BX,offset buffa         ;jmp setdma to complete call
1536  =                     ;
1537  =                     setdma:
1538  =                             ;BX=,dma address to set (i.e., buffa or dmaad)
1539  =0A18 8B0F                    MOV     CX,[bx]                 ;parameter ready
1540  =0A1A E90708      1224        JMP     setdmf
1541  =                     ;
1542  =                     ;
1543  =                     dirtouser:
1544  =                             ;copy the directory entry to the user buffer
1545  =                             ;after call to search or searchn by user code
1546  =0A1D 8B162A10                MOV     DX,buffa                ;source is directory buffer
1547  =0A21 8B1EA011                MOV     BX,dmaad                ;destination is user dma addr.
1548  =0A25 B180                    MOV     CL,recsiz               ;copy entire record
1549  =0A27 06                      push    es                      ;move to user segment
1550  =0A28 8E069D11                mov     es,parametersegment
1551  =0A2C E87EFD      07AD        call    move
1552  =0A2F 07                      pop     es
1553  =0A30 C3                      ret
1554  =                                                             ;ret
1555  =                     ;
1556  =                     endofdir:
1557  =                             ;return zero flag if at end of director, non zero
1558  =                             ;if not at end (end of dir if dcnt = 0ffffh)
1559  =0A31 BBA611                  mov     BX,offset dcnt
1560  =0A34 8A07                    MOV     AL,[bx]                 ;may be 0ffh
1561  =0A36 43                      INC     BX
1562  =0A37 3A07                    CMP     AL,[bx]                 ;low(dcnt) = high(dcnt)?
1563  =0A39 7502        0A3D        jnz     ret8                    ;return non zero if different
1564  =                                                             ;high and low the same,= 0ffh?
1565  =0A3B FEC0                    INC     AL                      ;0ffh becomes 00 if so
1566  =0A3D C3              ret8:   RET
1567  =                     ;
1568  =                     setenddir:
1569  =                             ;set dcnt to the end of the directory
1570  =0A3E BBFFFF                  MOV     BX,enddir
1571  =0A41 891EA611                MOV     dcnt,bx
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1572
1573  =0A45 C3                          RET
1574  =                         ;
1575  =                         rddir:
1576  =                                 ;read next directory entry, with CL=true if initializing
1577  =0A46 8B163910                    MOV     DX,dirmax           ;in preparation for subtract
1578  =0A4A 8B1EA611                    MOV     BX,dcnt
1579  =0A4E 43                          INC     BX
1580  =0A4F 891EA611                    MOV     dcnt,bx             ;dcnt=dcnt+1
1581  =                                                             ;continue while dirmax >= dcnt
1582  =                                                             ;(dirmax-dcnt no cy)
1583  =0A53 E864FF      09BA            CALL    subdh               ;DX-BX
1584  =0A56 7302        0A5A            JAE     rddir0
1585  =                                                             ;yes, set dcnt to end
1586  =                                                             ;of directory
1587  =0A58 EBE4        0A3E            jmps    setenddir
1588  =                         ;               ret
1589  =                         rddir0:
1590  =                                 ;not at end of directory, seek next element
1591  =                                                             ;CL=initialization flag
1592  =0A5A A0A611                      MOV     AL,ldcnt
1593  =0A5D 2403                        AND     AL,dskmsk           ;low(dcnt) and dskmsk
1594  =0A5F 51                          push    cx
1595  =0A60 B105                        MOV     cl,fcbshf           ;to multiply by fcb size
1596  =0A62 D2E0                        shl     al,cl
1597  =0A64 59                          pop     cx
1598  =                                                             ;A = (low(dcnt) and dskmsk)
1599  =                                                             ;shl fcbshf
1600  =0A65 A25610                      MOV     dptr,AL             ;ready for next dir operation
1601  =0A68 0AC0                        OR      AL,AL
1602  =0A6A 750B        0A77            jnz     ret71               ;return if not a new record
1603  =0A6C 51                          PUSH    CX                  ;save initialization flag CL
1604  =0A6D E8CCFD      083C            CALL    seekdir             ;seek proper record
1605  =0A70 E88BFF      09FE            CALL    rddirbuf            ;read the directory record
1606  =0A73 59                          POP     CX                  ;recall initialization flag
1607  =0A74 E94CFF      09C3            JMP     checksum            ;checksum the directory elt
1608  =                                                             ;ret
1609  =                         ;
1610  =                         ;
1611  =                                 if      not loader
1612  =                                 endif
1613  =0A77 C3                  ret71:  ret
1614  =                         ;
1615  =                                 if      not loader
1616  =                                 endif
1617  =                         ;
1618  =                         ;************** end BDOS filesystem part 1 ***************
1619                            end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1620
1621  =                     eject ! include bdosf2.a86     ; file system part 2
1622  =                     ;****************** BDOS file system part 2 ******************
1623  =                     ;
1624  =                     scndm:
1625  =                             ;scan the disk map addressed by dptr for non-zero
1626  =                             ;entries, the allocation vector entry corresponding
1627  =                             ;to a non-zero entry is set to the value of CL (0,1)
1628  =                             ;BX addresses the beginning of the directory entry
1629  =                     ;
1630  =0A78 E806FF      0981        CALL    getdptra                ;BX = buffa + dptr
1631  =0A7B 83C310                  ADD     BX,dskmap               ;BX = diskmap address
1632  =0A7E 51                      PUSH    CX                      ;save the 0/1 bit to set
1633  =0A7F B111                    MOV     CL,fcblen-dskmap+1      ;size of single byte diskmap+1
1634  =                     scndm0:
1635  =                                                             ;loop for each diskmap entry
1636  =0A81 5A                      POP     DX                      ;recall bit parity
1637  =0A82 FEC9                    DEC     CL
1638  =0A84 7501        0A87        JNZ     L_41
1639  =0A86 C3                      RET
1640  =                     L_41:                                   ;all done scanning?
1641  =                                                             ;no, get next entry for scan
1642  =0A87 52                      PUSH    DX                      ;replace bit parity
1643  =0A88 803E4A1000              cmp     single,0
1644  =0A8D 7408        0A97        JZ      scndm1
1645  =                                                             ;single byte scan operation
1646  =0A8F 51                      PUSH    CX                      ;save counter
1647  =0A90 53                      PUSH    BX                      ;save map address
1648  =0A91 8A0F                    MOV     CL,[bx]
1649  =0A93 B500                    MOV     CH,0                    ;CX=block#
1650  =0A95 EB07        0A9E        jmps    scndm2
1651  =                     scndm1:
1652  =                                                             ;double byte scan operation
1653  =0A97 FEC9                    DEC     CL                      ;count for double byte
1654  =0A99 51                      PUSH    CX                      ;save counter
1655  =0A9A 8B0F                    MOV     cx,[bx]                 ;CX=block
1656  =0A9C 43                      INC     BX
1657  =0A9D 53                      PUSH    BX                      ;save map address
1658  =                     scndm2:
1659  =                                                             ;arrive here with CX=block#,
1660  =                                                             ;DL= 0/1
1661  =0A9E 0BC9                    OR      cx,cx                   ;skip if = 0000
1662  =0AA0 7408        0AAA        JZ      scan3
1663  =0AA2 8B1E3710                MOV     BX,maxall               ;check invalid index
1664  =0AA6 3BD9                    cmp     bx,cx
1665  =0AA8 7200        0AAA        JNAE    L_42
1666  =                             if      not loader
1667  =                             endif
1668  =                     L_42:
1669  =
1670  =                     scan3:
1671  =0AAA 5B                      POP     BX
1672  =0AAB 43                      INC     BX
```

```
;
;bit set to 0/1
;
;to next bit position
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1673
1674 =0AAC 59                          POP     CX                      ;recall counter
1675 =0AAD EBD2          0A81          jmps    scndm0                  ;for another item
1676 =                          ;
1677 =                          initialize:
1678 =                                  ;initialize the current disk
1679 =                                  ;lret = false ;set to true if $ file exists
1680 =                                  ;compute the length of the allocation vector - 2
1681 =0AAF 8B1E3710                      MOV     BX,maxall
1682 =0AB3 B103                          MOV     CL,3                    ;perform maxall/8
1683 =0AB5 D3EB                          shr     bx,cl
1684 =0AB7 43                           INC     BX
1685 =0AB8 8BCB                          mov     cx,bx
1686 =0ABA 8B1E3010                      MOV     BX,alloca               ;base of allocation vector
1687 =                                  ;fill the allocation vector
1688 =                                  ;with zeros
1689 =                          init0:
1690 =0ABE C60700                        MOV     b[bx],0
1691 =0AC1 43                           INC     BX                      ;alloc(i)=0
1692 =0AC2 E2FA          0ABE          loop    init0
1693 =                                  ;set the reserved space for
1694 =                                  ;the directory
1695 =0AC4 8B163B10                      MOV     DX,dirblk
1696 =0AC8 8B1E3010                      MOV     BX,alloca               ;BX=.alloc()
1697 =0ACC 8917                          MOV     [bx],dx                 ;sets reserved directory blks
1698 =                                  ;allocation vector initialized
1699 =0ACE E845FD        0816          CALL    home                    ;home disk
1700 =                                  ;cdrmax = 3 (scans at least
1701 =                                  ;one directory record)
1702 =0AD1 8B1E2410                      MOV     BX,cdrmaxa
1703 =0AD5 C7070300                      mov     w[bx],3
1704 =                                  ;cdrmax = 0000
1705 =0AD9 E862FF        0A3E          CALL    setenddir               ;dcnt = enddir
1706 =                                  ;read directory entries and
1707 =                                  ;check for allocated storage
1708 =                          init2:
1709 =0ADC B1FF                          MOV     CL,true
1710 =0ADE E865FF        0A46          CALL    rddir
1711 =0AE1 E84DFF        0A31          CALL    endofdir
1712 =0AE4 743B          0B21          jz      ret8b                   ;rz
1713 =                                  ;not end of directory,
1714 =                                  ;valid entry ?
1715 =0AE6 E898FE        0981          CALL    getdptra                ;BX = buffa + dptr
1716 =0AE9 803FE5                       cmp     b[bx],empty
1717 =0AEC 74EE          0ADC          JZ      init2                   ;go get another item
1718 =                                  ;not empty,user code the same?
1719 =0AEE A0A511                       MOV     AL,usrcode
1720 =0AF1 3A07                         CMP     AL,[bx]
1721 =0AF3 750C          0B01          JNZ     pdollar
1722 =
1723 =                                  ;same user code,
1724 =0AF5 43                           INC     BX                      ;check for $ submit
1725 =0AF6 8A07                         MOV     AL,[bx]                 ;first character
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1726
1727  =0AF8 2C24                    SUB     AL,'$'                  ;dollar file?
1728  =0AFA 7505            0B01    JNZ     pdollar
1729  =                                                             ;$ file found, mark in lret
1730  =0AFC FEC8                    DEC     AL
1731  =0AFE A29B11                  MOV     lret,al                 ;lret = 255
1732  =                     pdollar:
1733  =                                                             ;now scan the disk map for
1734  =                                                             ;allocated blocks
1735  =0B01 B101                    MOV     CL,1                    ;set to allocated
1736  =0B03 E872FF          0A78    CALL    scndm
1737  =0B06 E8A8FE          09B1    CALL    setcdr                  ;set cdrmax to dcnt
1738  =0B09 EBD1            0ADC    jmps    init2                   ;for another entry
1739  =                     ;
1740  =                     cpydirloc:
1741  =                             ;copy directory location to lre following
1742  =                             ;delete, rename, ... ops
1743  =0B0B A04610                  MOV     AL,dirloc
1744  =0B0E E977FC          0788    JMP     staret
1745  =                     ;       ret
1746  =                     ;
1747  =                     compext:
1748  =                             ;compare extent# in AL with that in CL, return nonzero
1749  =                             ;if they do not match
1750  =0B11 51                      PUSH    CX                      ;save CL's original value
1751  =0B12 8A2E3610                MOV     ch,extmsk
1752  =0B16 F6D5                    NOT     ch
1753  =                                                             ;CH has negated form of
1754  =                                                             ;extent mask
1755  =0B18 22CD                    and     cl,ch                   ;low bits removed from CL
1756  =0B1A 22C5                    AND     AL,CH                   ;low bits removed from AL
1757  =0B1C 2AC1                    SUB     AL,CL
1758  =0B1E 241F                    AND     AL,maxext               ;set flags
1759  =0B20 59                      POP     CX                      ;restore original values
1760  =0B21 C3              ret8b:  RET
1761  =                     ;
1762  =                     ;
1763  =                     searchi:
1764  =                     ;       search initialization
1765  =0B22 8B1E9911                mov     bx,info
1766  =0B26 891E5910                mov     srcha,bx        ;searcha = info
1767  =                     searchi1:
1768  =0B2A C6064610FF              mov     dirloc,0ffh
1769  =0B2F 880EA811                mov     srchl,cl
1770  =0B33 C3                      ret
1771  =                     search:
1772  =                     srch:
1773  =                                                             ;search for directory element
1774  =                                                             ;of lenght
1775  =0B34 E8EBFF          0B22    call    searchi
1776  =0B37 E804FF          0A3E    CALL    setenddir               ;dcnt = enddir
1777  =0B3A E8D9FC          0816    CALL    home                    ;to start at the beginning
1778  =                                                             ;(drop through to searchn)
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1779
1780  =                     ;
1781  =                     searchn:
1782  =                     srchn:
1783  =                             ;search for the next directory element, assuming
1784  =                             ;a previous call on search which sets searcha and searchhl
1785  =                             ;
1786  =0B3D C606651000              mov     user_zero_pass,false
1787  =                     ;
1788  =0B42 B100                    MOV     CL,false
1789  =0B44 E8FFFE      0A46        CALL    rddir           ;read next dir element
1790  =0B47 E8E7FE      0A31        CALL    endofdir
1791  =0B4A 7503        0B4F        jnz     srchn1
1792  =                     ;
1793  =                     jmp_srchfin:
1794  =0B4C E98000      0BCF        jmp      srchfin
1795  =                     ;
1796  =                     srchn1:
1797  =                                                     ;skip to end if so
1798  =                                                     ;not end of directory,
1799  =                                                     ;scan for match
1800  =0B4F 8B165910                MOV     dx,srcha        ;DX=beginning of user fcb
1801  =0B53 8BF2                    MOV     SI,DX
1802  =0B55 AC                      LODS    al              ;first character
1803  =0B56 3CE5                    CMP     AL,empty        ;keep scanning if empty
1804  =0B58 7407        0B61        JZ      srchnext
1805  =                                                     ;not empty, may be end of
1806  =                                                     ;logical directory
1807  =0B5A 52                      PUSH    DX              ;save search address
1808  =0B5B E848FE      09A6        CALL    compcdr         ;past logical end?
1809  =0B5E 5A                      POP     DX              ;recall address
1810  =0B5F 73EB        0B4C        jnb     jmp_srchfin
1811  =                                                     ;artificial stop
1812  =                     srchnext:
1813  =0B61 E81DFE      0981        CALL    getdptra        ;BX = buffa+dptr
1814  =0B64 8A0EA811                mov     cl,srcl         ;length of search to CL
1815  =0B68 B500                    MOV     CH,0            ;CH counts up, CL counts down
1816  =                     srchloop:
1817  =0B6A 0AC9                    or      cl,cl
1818  =0B6C 7438        0BA6        JZ      endsearch
1819  =0B6E 8BF2                    MOV     SI,DX
1820  =0B70 AC                      LODS    al
1821  =0B71 3C3F                    CMP     AL,'?'
1822  =0B73 7429        0B9E        JZ      srchok          ;? matches all
1823  =                                                     ;scan next character if not
1824  =                                                     ;ubytes
1825  =0B75 80FD0D                  CMP     ch,ubytes
1826  =0B78 7424        0B9E        JZ      srchok
1827  =                                                     ;not the ubytes field,
1828  =                                                     ;extent field
1829  =0B7A 80FD0C                  CMP     ch,extnum       ;may be extent field
1830  =0B7D 7408        0B87        JZ      srchext         ;skip to search extent
1831  =0B7F 2A07                    SUB     AL,[bx]
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1832
1833  =0B81 247F                    AND     AL,7fh                  ;mask-out flags/extent modulus
1834  =0B83 7533          0BB8      jnz     srchnm
1835  =0B85 EB17          0B9E      jmps    srchok                  ;matched character
1836  =                         srchext:
1837  =                                                             ;A has fcb character
1838  =                                                             ;attempt an extent # match
1839  =0B87 51                      PUSH    CX                      ;save counters
1840  =0B88 8A0F                    MOV     CL,[bx]                 ;directory character to c
1841  =0B8A E884FF        0B11      CALL    compext                 ;compare user/dir c'ar
1842  =0B8D 59                      POP     CX                      ;recall counters
1843  =0B8E 9F                      lahf
1844  =0B8F F6066510FF              test    user_zero_pass,true
1845  =0B94 7545          0BDB      jnz     save_dcnt_pos2          ;save position o dcnt
1846  =                         ;
1847  =                             disable search under user zero if any fcb found under t'is user
1848  =                         ;
1849  =0B96 C606621000              mov     search_user0,false
1850  =0B9B 9E                      sahf
1851  =0B9C 759F          0B3D      jnz     srchn                   ;skip if no match
1852  =                         srchok:
1853  =                                                             ;current character matches
1854  =0B9E 42                      INC     DX
1855  =0B9F 43                      INC     BX
1856  =0BA0 FEC5                    INC     CH
1857  =0BA2 FEC9                    DEC     CL
1858  =0BA4 EBC4          0B6A      jmps    srchloop
1859  =                         endsearch:
1860  =                             ;entire name matches, return dir position
1861  =0BA6 C606461000              mov     dirloc,0h               ;mark as found
1862  =0BAB A0A611                  MOV     AL,ldcnt
1863  =0BAE 2403                    AND     AL,dskmsk               ;return code 0-3
1864  =0BB0 A27B11                  MOV     lret,al
1865  =                                                             ;lret = low(dcnt) and 11b
1866  =0BB3 8AE8                    mov     ch,al
1867  =0BB5 FEC5                    inc     ch                      ;set the zero flag
1868  =0BB7 C3                ret9: RET
1869  =                         ;
1870  =                         srchnm:
1871  =0BB8 0AED                    or      ch,ch           ;search 'NO' match routine
1872  =0BBA 7530          0BEC      jnz     searchn_jmp
1873  =0BBC F607FF                  test    b[bx],true
1874  =0BBF 7528          0BEC      jnz     searchn_jmp
1875  =0BC1 F6066210FF              test    search_user0,true
1876  =0BC6 7424          0BEC      jz      searchn_jmp
1877  =0BC8 C6066510FF              mov     user_zero_pass,true
1878  =0BCD EBCF          0B9E      jmps    srchok
1879  =                         ;
1880  =                         srchfin:
1881  =                             ;end of directory, or empty nam
1882  =0BCF E836CFE       0A3E      CALL    setenddir               ;may be artifical end
1883  =0BD2 B0FF                    MOV     AL,255
1884  =0BD4 8AE8                    mov     ch,al
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1885
1886 =0BD6 FEC5                    inc     ch                  ;zero flag set on unsuccessful
1887 =0BD8 E9ADFB        0788      JMP     staret
1888 =                         ;
1889 =                         ;
1890 =                         save_dcnt_pos2:
1891 =0BDB 9E                          sahf
1892 =0BDC 750E          0BEC          jne     searchn_jmp
1893 =0BDE 43                          inc     bx
1894 =0BDF 803F00                      cmp     b[bx],0         ;check for modnum=0 under user 0
1895 =0BE2 7508          0BEC          jne     searchn_jmp
1896 =0BE4 8B36A611                    mov     si,dcnt
1897 =0BE8 89366310                    mov     xdcnt,si
1898 =                         searchn_jmp:
1899 =0BEC E94EFF        0B3D          jmp     searchn
1900 =                         ;
1901 =                                 if      not loader
1902 =                                 endif
1903 =                         ;
1904 =                                 if      not loader
1905 =                                 endif
1906 =                         ;
1907 =                         ;
1908 =                         ;
1909 =                         copyfcb:
1910 =                                                         ;copy the entire FCB
1911 =0BEF B100                        MOV     CL,0
1912 =0BF1 B220                        MOV     DL,fcblen       ;start at 0, to fcblen-1
1913 =                                                         ;       jmp copydir
1914 =                         ;
1915 =                         copydir:
1916 =                                 ;copy fcb information starting
1917 =                                 ;into the currently addressed directory entry
1918 =0BF3 52                          PUSH    DX              ;save length for later
1919 =0BF4 B500                        MOV     CH,0            ;double index to CX
1920 =0BF6 8B169911                    MOV     dx,info         ;BX = source for data
1921 =0BFA 03D1                        ADD     dx,CX           ;DX=.fcb(CL), source for copy
1922 =0BFC E882FD        0981          CALL    getdptra        ;BX=.buff(dptr), destination
1923 =0BFF 59                          POP     CX              ;DX=source, BX=dest, C=length
1924 =0C00 E8AAFB        07AD          CALL    move            ;data moved
1925 =                         seekcopy:
1926 =                                 ;enter from close to seek and copy current element
1927 =0C03 E836FC        033C          CALL    seekdir         ;to the directory element
1928 =0C06 E9E8FD        09F1          JMP     wrdir           ;write the directory element
1929 =                                                         ;ret
1930 =                         ;
1931 =                         ;
1932 =                                 if      not loader
1933 =                                 endif
1934 =                         ;
1935 =                                 if      loader
1936 =0C09 C3                  ret10:  ret
1937 =                                 endif
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1938
1939 =                    ;
1940 =                    ;
1941 =                    ;
1942 =                    indicators:
1943 =                            ;set file indicators for current fcb
1944 =0C0A B10C                   MOV     CL,extnum
1945 =0C0C E825FF      0B34       CALL    srch                ;through file type
1946 =                    indic0:
1947 =0C0F E81FFE      0A31       CALL    endofdir
1948 =0C12 74F5        0C09       jz      ret10               ;stop at end of dir
1949 =                                                       ;not end of directory,
1950 =                                                       ;continue to change
1951 =0C14 B100                   MOV     CL,0
1952 =0C16 B20C                   MOV     DL,extnum           ;copy name
1953 =0C18 E8D8FF      0BF3       CALL    copydir
1954 =0C1B E81FFF      0B3D       CALL    srchn
1955 =0C1E EBEF        0C0F       jmps    indic0
1956 =                    ;
1957 =                    open:
1958 =                            ;search for the directory entry, copy to FCB
1959 =0C20 B10F                   MOV     CL,namlen
1960 =0C22 E80FFF      0B34       CALL    srch
1961 =0C25 E809FE      0A31       CALL    endofdir
1962 =                    open1:
1963 =0C28 74DF        0C09       jz      ret10               ;return with lret=255 if end
1964 =                                                       ;not end of directory,copy fcb
1965 =                    opencopy:
1966 =                                                       ;(referenced below to copy fcb
1967 =0C2A E8AAFC      08D7       CALL    getexta
1968 =
1969 =0C2D 50                     PUSH    AX
1970 =0C2E 53                     PUSH    BX                  ;save extent#
1971 =0C2F E84FFD      0981       CALL    getdptra
1972 =0C32 87DA                   XCHG    BX,DX               ;DX = .buff(dptr)
1973 =0C34 8B1E9911               MOV     BX,info             ;BX=.fcb(0)
1974 =0C38 B120                   MOV     CL,nxtrec           ;length of move operation
1975 =0C3A 52                     PUSH    DX                  ;save .buff(dptr)
1976 =0C3B E836FB      07AD       CALL    move                ;from .buff(dptr) to .fcb(0)
1977 =                                                       ;note that entire fcb is
1978 =                                                       ;copied, including indicators
1979 =0C3E E85DFD      099E       CALL    setfwf              ;sets file write flag
1980 =0C41 5A                     POP     DX
1981 =0C42 BB0C00                 MOV     BX,extnum
1982 =0C45 03DA                   ADD     BX,DX               ;bx=.buff(dptr+extnum)
1983 =0C47 8A0F                   mov     cl,[bx]
1984 =
1985 =0C49 BB0F00                 MOV     BX,reccnt
1986 =0C4C 03DA                   ADD     BX,DX
1987 =0C4E 8A2F                   mov     ch,[bx]
1988 =
1989 =0C50 5B                     POP     BX
1990 =0C51 58                     POP     AX
```

CP/M 86 V.1.1 - vol I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP81-162

```
1991
1992 =0C52 8807                    MOV     [bx],al             ;restore extent number
1993 =                        ;
1994 =                        ;BX = .user extent#, CH = dir rec cnt, CL = dir extent#
1995 =                        ;if user ext < dir ext then user := 128 records
1996 =                        ;if user ext = dir ext then user := dir records
1997 =                        ;if user ext > dir ext then user := 0 records
1998 =                        ;
1999 =0C54 3A0F                    CMP     cl,[bx]
2000 =0C56 8AC5                    MOV     AL,CH               ;ready dir reccnt
2001 =0C58 7406          0C60      JZ      openrcnt            ;if same, user gets dir reccnt
2002 =0C5A B000                    MOV     AL,0
2003 =0C5C 7202          0C60      JB      openrcnt            ;user is larger
2004 =0C5E B080                    MOV     AL,128              ;directory is larger
2005 =                        openrcnt:                       ;A has record count to fill
2006 =0C60 8B1E9911                MOV     BX,info
2007 =0C64 8A0F00                  MOV     DX,reccnt
2008 =0C67 03DA                    ADD     BX,DX
2009 =0C69 8807                    MOV     [bx],al
2010 =0C6B C3                 ret11: RET
2011 =                        ;
2012 =                        mergezero:
2013 =                                                        ;BX = .fcb1(i), DX = .fcb2(i),
2014 =                                                        ;if fcb1(i) = 0
2015 =                                                        ;then fcb1(i) := fcb2(i)
2016 =0C6C 833F00                  cmp     w[bx],0
2017 =0C6F 75FA          0C6B      jnz     ret11               ;rnz
2018 =0C71 8BF2                    MOV     SI,DX
2019 =0C73 AD                      LODS    ax
2020 =0C74 8907                    mov     [bx],ax
2021 =0C76 C3                 ret12: RET
2022 =                        ;
2023 =                        close:
2024 =                              ;locate the directory element and re-write it
2025 =0C77 33C0                    XOR     AX,AX
2026 =0C79 A29B11                  MOV     lret,al
2027 =0C7C A3A611                  MOV     dcnt,ax
2028 =0C7F E8BEFC        0940      CALL    nowrite
2029 =0C82 75F2          0C76      jnz     ret12               ;skip close if r/o disk
2030 =                                                        ;check file write flag -
2031 =                                                        ;0 indicates written
2032 =0C84 E806FD        098D      CALL    getmodnum           ;fcb(modnum) in A
2033 =0C87 2480                    AND     AL,fwfmsk
2034 =0C89 75EB          0C76      jnz     ret12               ;return if bit remains set
2035 =0C8B B10F                    MOV     CL,namlen
2036 =0C8D E8A4FE        0B34      CALL    srch                ;locate file
2037 =0C90 E89EFD        0A31      CALL    endofdir
2038 =0C93 74E1          0C76      jz      ret12               ;return if not found
2039 =                                                        ;merge the disk map at info
2040 =0C95 B91000                  MOV     CX,dskmap
2041 =0C98 E8E6FC        0981      CALL    getdptra
2042 =0C9B 03D9                    ADD     BX,CX
2043 =0C9D 87DA                    XCHG    BX,DX               ;DX is .buff(dptr+16)
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2044
2045  =0C9F 8B1E9911              MOV     BX,info
2046  =0CA3 03D9                  ADD     BX,CX               ;dx=.buff(dptr+16),
2047  =                                                       ;bx=.fcb(16)
2048  =0CA5 B110                  MOV     CL,(fcblen-dskmap)  ;length of single byte dm
2049  =                       merge0:
2050  =0CA7 803E4A1000            cmp     single,0
2051  =0CAC 741B        0CC9      JZ      merged              ;skip to double
2052  =                       ;
2053  =                           ;this is a single byte map
2054  =                           ;if fcb(i) = 0 then fcb(i) = buff(i)
2055  =                           ;if buff(i) = 0 then buff(i) = fcb(i)
2056  =                           ;if fcb(i) <> buff(i) then error
2057  =                       ;
2058  =0CAE 8A07                  MOV     AL,[bx]
2059  =0CB0 0AC0                  OR      AL,AL
2060  =0CB2 8BF2                  MOV     SI,DX
2061  =0CB4 AC                    LODS    al
2062  =0CB5 7502        0CB9      JNZ     fcbnzero
2063  =                                                       ;fcb(i) = 0
2064  =0CB7 8807                  MOV     [bx],al             ;fcb(i) = buff(i)
2065  =                       fcbnzero:
2066  =0CB9 0AC0                  OR      AL,AL
2067  =0CBB 7506        0CC3      JNZ     buffnzero
2068  =                                                       ;buff(i) = 0
2069  =0CBD 8A07                  MOV     AL,[bx]
2070  =0CBF 8BFA                  MOV     DI,DX
2071  =0CC1 FC                    cld
2072  =0CC2 AA                    stos    al                  ;buff(i)=fcb(i)
2073  =                       buffnzero:
2074  =0CC3 3A07                  CMP     AL,[bx]
2075  =0CC5 7547        0D0E      jnz     mergerr             ;fcb(i) = buff(i)?
2076  =0CC7 EB16        0CDF      jmps    dmset               ;if merge ok
2077  =                       merged:
2078  =                                                       ;this is a double byte merge
2079  =0CC9 E8A0FF      0C6C      CALL    mergezero           ;buff = fcb if buff 0000
2080  =0CCC 87DA                  XCHG    BX,DX
2081  =0CCE E89BFF      0C6C      CALL    mergezero
2082  =0CD1 87DA                  XCHG    BX,DX               ;fcb = buff if fcb 0000
2083  =                                                       ;they should be identical
2084  =                                                       ;at this point
2085  =0CD3 8BF2                  MOV     SI,DX
2086  =0CD5 8B04                  mov     ax,[si]
2087  =0CD7 3B07                  cmp     ax,[bx]
2088  =0CD9 7533        0D0E      jnz     mergerr
2089  =0CDB 42                    inc     dx
2090  =0CDC 43                    inc     bx
2091  =                                                       ;merge ok for this pair
2092  =0CDD FEC9                  DEC     CL                  ;extra count for double byte
2093  =                       dmset:
2094  =0CDF 42                    INC     DX
2095  =0CE0 43                    INC     BX
2096  =0CE1 FEC9                  DEC     CL
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2097
2098  =0CE3 75C2          0CA7          JNZ     merge0              ;for more
2099  =                                                            ;end of disk map merge,
2100  =                                                            ;check record count
2101  =                                                            ;DX = .buff(dptr)+32,
2102  =                                                            ;BX = .fcb(32)
2103  =0CE5 B9ECFF                      MOV     CX,-(fcblen-extnum)
2104  =0CE8 03D9                        ADD     BX,CX
2105  =0CEA 87DA                        XCHG    BX,DX
2106  =0CEC 03D9                        ADD     BX,CX
2107  =                                                            ;DX = .fcb(extnum),
2108  =                                                            ;BX = .buff(dptr+extnum)
2109  =0CEE 8BF2                        MOV     SI,DX
2110  =0CF0 AC                          LODS    al                  ;current user extent number
2111  =                                                            ;if fcb(ext) >= buff(fcb) then
2112  =                                                            ;buff(ext) := fcb(ext),
2113  =                                                            ;buff(rec) := fcb(rec)
2114  =0CF1 3A07                        CMP     AL,[bx]
2115  =0CF3 7211          0D06          JB      endmerge
2116  =                                                            ;fcb extent number >=
2117  =                                                            ;dir extent number
2118  =0CF5 8807                        MOV     [bx],al             ;buff(ext) = fcb(ext)
2119  =                                                            ;update dir. rec count field
2120  =0CF7 B90300                      MOV     CX,(reccnt-extnum)
2121  =0CFA 03D9                        ADD     BX,CX
2122  =0CFC 87DA                        XCHG    BX,DX
2123  =0CFE 03D9                        ADD     BX,CX
2124  =                                                            ;DX=.buff(reccnt),
2125  =                                                            ;BX=.fcb(reccnt)
2126  =0D00 8A07                        MOV     AL,[bx]
2127  =0D02 8BFA                        MOV     DI,DX
2128  =0D04 FC                          cld
2129  =0D05 AA                          stos    al                  ;buff(reccnt)=fcb(reccnt)
2130  =                         endmerge:
2131  =0D06 C6064310FF                  MOV     fcbcopied,true      ;mark as copied
2132  =0D0B E9F5FE        0C03          JMP     seekcopy            ;ok to 'wrdir' here -
2133  =                                                            ;1.4 compatible
2134  =                                                            ; ret
2135  =                         mergerr:
2136  =                                                            ;elements did not merge ok
2137  =0D0E FE0E9B11                    dec     lret                ;=255 non zero flag set
2138  =0D12 C3                  ret13:  RET
2139  =                         ;
2140  =                         ;
2141  =                                 if      not loader
2142  =                                 endif
2143  =                         ;
2144  =                         ;
2145  =                         openreel:
2146  =                                 ;close the current extent, and open the next one
2147  =                                 ;if possible.  RMF is true if in read mode.
2148  =0D13 32C0                        XOR     AL,AL
2149  =0D15 A24310                      MOV     fcbcopied,AL        ;set true if actually copied
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. Box 579
PACIFIC GROVE, CA 93950
SER. #

```
2150
2151  =0D18 E85CFF      0C77        CALL    close               ;close current extent
2152  =                                                         ;lret remains at enddir
2153  =                                                         ;if we cannot open next ext
2154  =0D1B E813FD      0A31        CALL    endofdir
2155  =0D1E 74F2        0D12        jz      ret13               ;return if end
2156  =                                                         ;increment extent number
2157  =0D20 8B1E9911                MOV     BX,info
2158  =0D24 83C30C                  ADD     BX,extnum           ;BX=.fcb(extnum)
2159  =0D27 8A07                    MOV     AL,[bx]
2160  =0D29 FEC0                    INC     AL
2161  =0D2B 241F                    AND     AL,maxext
2162  =0D2D 8807                    MOV     [bx],al             ;fcb(extnum)=++1
2163  =0D2F 740F        0D40        JZ      openmod             ;move to next module if 0
2164  =                                                         ;may be in same extent group
2165  =0D31 8AE8                    MOV     CH,AL
2166  =0D33 A03610                  MOV     AL,extmsk
2167  =0D36 22C5                    AND     AL,CH
2168  =                                                         ;if 0, then not in same group
2169  =0D38 22064310                and     al,fcbcopied        ;true if fcb was copied to dir
2170  =                                                         ;puts 0 in AL if not written
2171  =0D3C 740D        0D4B        JZ      openr0              ;go to next physical extent
2172  =                                                         ;result is non zero,
2173  =                                                         ;so we must be in same extent
2174  =0D3E EB1C        0D5C        jmps    openr1              ;to copy fcb info
2175  =                       openmod:
2176  =                               ;extent number overflow, go to next module
2177  =0D40 83C302                  ADD     BX,(modnum-extnum)  ;bx=.fcb(modnum)
2178  =0D43 FE07                    INC     b[bx]               ;fcb(modnum)=++1
2179  =                                                         ;module number incremented,
2180  =                                                         ;check for overflow
2181  =0D45 8A07                    MOV     AL,[bx]
2182  =0D47 240F                    AND     AL,maxmod           ;mask high order bits
2183  =0D49 741C        0D67        JZ      openerr             ;cannot overflow to 0
2184  =                                                         ;otherwise, ok to continue
2185  =                                                         ;with new module
2186  =                       openr0:
2187  =0D4B B10F                    MOV     CL,namlen
2188  =0D4D E8E4FD      0B34        CALL    srch                ;next extent found?
2189  =0D50 E8DEFC      0A31        CALL    endofdir
2190  =0D53 7507        0D5C        JNZ     openr1
2191  =                                                         ;end of file encountered
2192  =0D55 A04410                  MOV     AL,rmf
2193  =0D58 FEC0                    INC     AL                  ;0ffh becomes 00 if read
2194  =0D5A 7408        0D67        JZ      openerr             ;sets lret = 1
2195  =                                                         ;try to extend current file
2196  =                               if      not loader
2197  =                               endif
2198  =                       ;
2199  =                       ;
2200  =                       openr1:
2201  =
2202  =0D5C E8CDFE      0C2A        CALL    opencopy
                                                          ;not end of file, open
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2203
2204  =                       openr2:
2205  =0D5F E88CFB    08EE            CALL    getfcb          ;set parameters
2206  =0D62 32C0                      XOR     AL,AL           ;
2207  =0D64 E721FA    0738            JMP     staret          ;lret = 0
2208  =                               ;                       ; ret
2209  =                       openerr:
2210  =                                                       ;cannot move to next extent
2211  =                                                       ;of this file
2212  =0D67 E822FA    078C            CALL    setlret1        ;lret = 1
2213  =0D6A E931FC    099E            JMP     setfwf          ;ensure that it will not
2214  =                                                       ;be closed
2215  =                                                       ;ret
2216  =                       ;
2217  =                       ;***************** end BDOS file system part 2 ****************
2218                          end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2219
2220  =                     eject ! include bdosf3.a86     ; file system part 3
2221  =                     ;********************* BDOS filesystem part 3 *****************
2222  =                     ;
2223  =                     ;
2224  =                     ;
2225  =                     sqread:
2226  =                             ;sequential disk read operation
2227  =0D6D C606471001              MOV     seqio,1
2228  =                                                     ;drop through to diskread
2229  =                     ;
2230  =                     diskread:                       ;(may enter from seqdiskread)
2231  =0D72 C6064410FF              MOV     rmf,true        ;read mode flag =
2232  =                                                     ;true (open$reel)
2233  =                                                     ;read the next record from
2234  =                                                     ;the current FCB
2235  =0D77 E874FB      08EE        CALL    getfcb          ;sets parameters for the read
2236  =0D7A A05010                  MOV     AL,lvrecord
2237  =0D7D 3A064E10                CMP     AL,rcount       ;vrecord-rcount
2238  =                                                     ;skip if rcount > vrecord
2239  =0D81 7218        0D9B        JB      recordok
2240  =                                                     ;not enough records in extent
2241  =                                                     ;record count must be 128
2242  =                                                     ;to continue
2243  =0D83 3C80                    CMP     AL,128          ;vrecord = 128?
2244  =0D85 7528        0DAF        JNZ     diskeof         ;skip if vrecord<>128
2245  =0D87 E889FF      0D13        CALL    openreel        ;go to next extent if so
2246  =0D8A E854FB      08E1        call    gtfcba          ;get address of next record
2247  =0D8D 33C0                    xor     ax,ax
2248  =0D8F A35010                  MOV     vrecord,ax       ;vrecord=00
2249  =0D92 8807                    mov     [bx],al         ;zero record number
2250  =                                                     ;now check for open ok
2251  =0D94 803E7B1100              cmp     lret,0          ;stop at eof
2252  =0D99 7514        0DAF        JNZ     diskeof
2253  =                     recordok:
2254  =                             ;arrive with fcb addressing a record to read
2255  =0D9B E807FB      08A5        CALL    index
2256  =                                                     ;error 2 if reading
2257  =                                                     ;unwritten data
2258  =                                                     ;(returns 1 to be compatible
2259  =                                                     ;with 1.4)
2260  =0D9E E813FB      08B4        CALL    alloct          ;arecord=0000?
2261  =0DA1 740C        0DAF        JZ      diskeof
2262  =                                                     ;record has been allocated,
2263  =                                                     ;read it
2264  =0DA3 E815FB      08BB        CALL    atran           ;arecord now a disk address
2265  =0DA6 E8A3FA      084C        CALL    seek            ;to proper track,sector
2266  =0DA9 E876FA      0822        CALL    rdbuff          ;to dma address
2267  =0DAC E95AFB      0907        JMP     setfcb          ;replace parameter
2268  =                     ;       ret
2269  =                     diskeof:
2270  =0DAF E9DAF9      078C        JMP     setlret1
2271  =                                                     ;lret = 1
                                                            ;ret
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2272
2273 =                       ;
2274 =                               if      not loader
2275 =                               endif
2276 =                       ;
2277 =                       ;
2278 =                       ;
2279 =                       ;
2280 =                       ;
2281 =                       slect:
2282 =                               ;select disk word ptr info for subsequent input or output ops
2283 =0DB2 8B1E2210                    MOV     BX,dlog
2284 =0DB6 8A0EA411                    MOV     cl,curdsk
2285 =0DBA D3CB                        ror     bx,cl
2286 =0DBC 53                         PUSH    BX
2287 =0DBD 87D3                       XCHG    BX,DX                   ;save it for test below,
2288 =                                                                ;send to seldsk
2289 =0DBF E8F7F9       07B9          CALL    selectdisk
2290 =0DC2 5B                         POP     BX                      ;recall dlog vector
2291 =0DC3 7503         0DC8          JNZ     L_67
2292 =0DC5 E8E0F9       07A8          CALL    selerror
2293 =                       L_67:                                    ;returns true if select ok
2294 =                                                                ;is the disk logged in?
2295 =0DC8 D0D8                       RCR     bl,1
2296 =0DCA 7267         0E33          jb      ret20                   ;rc
2297 =                                                                ;return if bit is set
2298 =                                                                ;disk not logged in,
2299 =                                                                ;set bit and initialize
2300 =0DCC 8B0E2210                    mov     cx,dlog
2301 =0DD0 E85FFB       0932          CALL    setcdisk
2302 =0DD3 891E2210                    MOV     dlog,BX                 ;dlog=set$cdisk(dlog)
2303 =0DD7 E9D5FC       0AAF          JMP     initialize
2304 =                                                                ;ret
2305 =                       ;
2306 =                       curselect:
2307 =0DDA A04810                      MOV     AL,linfo
2308 =0DDD A2A411                      mov     curdsk,al               ;curdsk=info
2309 =0DE0 EBD0         0DB2          jmps    slect
2310 =                                                                ;ret
2311 =                       parsave33r:
2312 =0DE2 E81FF7       0504          call    parsave33
2313 =0DE5 EB03         0DEA          jmps    reselect
2314 =                       ;
2315 =                       parsave36r:
2316 =0DE7 E81FF7       0509          CALL    parsave36
2317 =                       ;
2318 =                       ;
2319 =                       reselect:
2320 =                               ;check current fcb to see if reselection necessary
2321 =0DEA 8B1E9911                    mov     bx,info                 ;examine attribute bits
2322 =0DEE C606611000                  mov     high_ext,0
2323 =0DF3 F6470880                    test    f8[bx],interfc_bit              ;fcb(8) and 80h
2324 =
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2325
2326  =0DF7 7409          0E02          jz      reselect0
2327  =0DF9 8067087F                    and     f8[bx],not_interfc_bit      ;change FCB copy
2328  =0DFD C606611060                  mov     high_ext,zero_readro        ;then high_ext = 60h
2329  =                           reselect0:
2330  =0E02 C6064B10FF                  MOV     resel,true          ;mark possible reselect
2331  =0E07 8B1E9911                    MOV     BX,info
2332  =0E0B 8A07                        MOV     AL,[bx]             ;drive select code
2333  =0E0D 241F                        AND     AL,00011111b        ;non zero is auto drive select
2334  =0E0F FEC8                        dec     AL                  ;drive code normalized to
2335  =                                                             ;0...30, or 255
2336  =0E11 A24810                      MOV     linfo,AL            ;save drive code
2337  =0E14 3C1E                        CMP     AL,30
2338  =0E16 7312          0E2A          JAE     noselect
2339  =                                                             ;auto select function,
2340  =                                                             ;save curdsk
2341  =0E18 A0A411                      MOV     AL,curdsk
2342  =0E1B A24C10                      MOV     olddsk,AL           ;olddsk=curdsk
2343  =0E1E 8A07                        MOV     AL,[bx]
2344  =0E20 A24D10                      MOV     fcbdsk,AL           ;save drive code
2345  =0E23 24E0                        AND     AL,11100000b
2346  =0E25 8807                        MOV     [bx],al             ;preserve hi bits
2347  =0E27 E8B0FF        0DDA          CALL    curselect
2348  =                           noselect:
2349  =                                                             ;set user code
2350  =0E2A A0A511                      MOV     AL,usrcode          ;0...31
2351  =0E2D 8B1E9911                    MOV     BX,info
2352  =0E31 0807                        or      b[bx],al
2353  =0E33 C3                    ret20:  RET
2354  =                           ;
2355  =                           ;
2356  =                           check_fcb:
2357  =0E34 803L611060                  cmp     high_ext,zero_readro    ;was file opened under user 0
2358  =0E39 7507          0E42          jne     xcheck_fcb
2359  =0E3B 8B1E9911                    mov     bx,info
2360  =0E3F C60700                      mov     b[bx],0h            ;zero the user number
2361  =                           xcheck_fcb:
2362  =0E42 C3                          ret
2363  =                           ;
2364  =                           ;
2365  =                           ;
2366  =                           ;
2367  =                           ;       individual function handlers
2368  =                           ;func12:
2369  =                           ;
2370  =                                   ;return version number
2371  =                           ;
2372  =                           ;
2373  =                           func13:
2374  =                           ;
2375  =                                   ;reset disk system - initialize to disk 0
2376  =                           ;
2377  =0E43 F7060210FFFF                  test    bioscs,0ffffh
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2378
2379 =0E49 7504          0E4F          jnz     func13a
2380 =0E4B 8C0E0210                    mov     bioscs,cs
2381 =                           func13a:
2382 =0E4F 33DB                       xor     bx,bx
2383 =0E51 891E2010                    MOV     rodsk,BX
2384 =0E55 891E2210                    MOV     dlog,BX
2385 =0E59 8A1EB711                    mov     bl,curdrvs          ;allow sys load from sys drive
2386 =0E5D 881EA411                    MOV     curdsk,bl           ;note that usrcode remains
2387 =                                                             ;unchanged
2388 =0E61 E94EFF        0DB2          JMP     slect
2389 =                                                             ;ret ;Jmp goback
2390 =                           ;
2391 = 0DDA                      func14  EQU     curselect           ;-----------------------------
2392 =                           ;
2393 =                                 ;select disk info
2394 =                           ;
2395 =                           ;
2396 =                           func15:
2397 =                           ;
2398 =                                 ;open file
2399 =                           ;
2400 =0E64 E87BFF        0DE2          CALL    parsave33r          ;copy FCB from user seg.
2401 =0E67 E82DFB        0997          CALL    clrmodnum           ;clear the module number
2402 =0E6A 803EA51100                  cmp     usrcode,0h          ;is current user # 0
2403 =0E6F 740B          0E7C          je      call_open
2404 =0E71 C7066410FE00                mov     found_in_zero,not_found ;flag to indicate file found
2405 =0E77 C6066210FF                  mov     search_user0,true   ;flat to indicate search user 0
2406 =                           ;
2407 =                           call_open:
2408 =0E7C E8A1FD        0C20          call    open
2409 =0E7F E83C00        0EBE          call    openx               ;returns iif unsuccessful
2410 =0E82 F6066210FF                  test    search_user0,true   ;do we search user 0?
2411 =                                                             ;don't search if any FCB
2412 =                                                             ;on other user
2413 =0E87 7501          0E8A          jnz     try_zero
2414 =                           ret30:
2415 =0E89 C3                         ret                         ;yes, open failed
2416 =                           try_zero:
2417 =0E8A C606621000                  mov     search_user0,false  ;reset switch
2418 =0E8F 813E6410FE00                cmp     found_in_zero,not_found ;file exists under user 0?
2419 =0E95 74F2          0E89          je      ret30
2420 =
2421 =0E97 8B1E6310                    mov     bx,xdcnt            ;reset dcnt
2422 =0E9B 80E3FC                      and     bl,0fch             ;set dcnt to last entry of
2423 =                                                             ;previous record
2424 =0E9E 4B                          dec     bx                  ;force read of dir record if change
2425 =0E9F 891EA611                    mov     dcnt,bx
2426 =0EA3 C606611060                  mov     high_ext,zero_readro ;set mode to read/only
2427 =                           open_user_zero:
2428 =0EA8 8B1E9911                    mov     bx,info
2429 =0EAC C60700                      mov     b[bx],0h            ;mov zero user # to fcb
2430 =0EAF B10F                        mov     cl,namlen
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2431
2432  =0EB1 E86EFC      0B22        call    searchi
2433  =0EB4 E836FC      0B3D        call    searchn
2434  =0EB7 E86EFD      0C28        call    open1
2435  =0EBA E80100      0EBE        call    openx
2436  =0EBD C3                      ret                         ;unsuccessful under user 0
2437  =                       ;
2438  =                       ;
2439  =                   openx:
2440  =0EBE E870FB      0A31        call    en_of_dir           ;open successful??
2441  =0EC1 74C6        0E89        jz      ret30
2442  =0EC3 5B                      pop     bx                  ;discard return address
2443  =0EC4 803E611060              cmp     high_ext,60h        ;is open mode read/only???
2444  =                                                         ;and user 0
2445  =0EC9 7512        0EDD        jne     openx0
2446  =0ECB 8B1E9911                mov     bx,info
2447  =0ECF 8A470A                  mov     al,10[bx]           ;byte t2 in FCB
2448  =0ED2 2480                    and     al,80h              ;check for system file
2449  =0ED4 7507        0EDD        jnz     openx0
2450  =0ED6 A26110                  mov     high_ext,al         ;open failed...not system
2451  =0ED9 B0FF                    mov     al,0ffh
2452  =0EDB EB4C        0F29        jmps    to_staret
2453  =                   openx0:
2454  =0EDD C3                      ret                         ;return from open function
2455  =                       ;
2456  =                                                         ;ret ;jmp goback
2457  =                       ;
2458  =                   func16:
2459  =                       ;
2460  =                               ;close file
2461  =                       ;
2462  =0EDE E801FF      0DE2        CALL    parsave33r
2463  =0EE1 E993FD      0C77        jmp     close
2464  =                                                         ;ret ;jmp goback
2465  =                       ;
2466  =                   func17:
2467  =                               ;search for first occurrence of a file
2468  =                       ;
2469  =0EE4 E81DF6      0504        call    parsave33
2470  =                       ;
2471  =0EE7 8B1E9D11                mov     bx,parametersegment ;save DS seg containing srcha
2472  =0EEB 891EAB11                mov     srcha_seg,bx
2473  =                       ;
2474  =0EEF B100                    MOV     CL,0                ;length assuming '?' true
2475  =0EF1 8B1E9911                mov     bx,info
2476  =0EF5 803F3F                  cmp     b[bx],'?'           ;no reselect if ?
2477  =0EF8 7410        0F0A        JZ      qselect             ;skip reselect if so
2478  =                                                         ;normal search
2479  =0EFA E8DAF9      08D7        CALL    getexta
2480  =0EFD 803F3F                  cmp     b[bx],'?'           ;check extent for '?'
2481  =0F00 7403        0F05        jz      all_modules
2482  =0F02 E892FA      0997        CALL    clrmodnum           ;module number zeroed
2483  =                   all_modules:
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
2484
2485  =0F05 E8E2FE      0DEA        CALL    reselect
2486  =0F08 B10F                    MOV     CL,namlen
2487  =                       qselect:
2488  =0F0A E827FC      0B34        CALL    srch
2489  =0F0D EB01        0F10        jmps    to_dirtouser        ;copy directory entry to user
2490  =                                                         ;ret ;jmp goback
2491  =                             ;
2492  =                             func18:
2493  =                             ;
2494  =                                     ;search for next occurrence of a file name
2495  =                                     if      loader
2496  =0F0F C3                              ret
2497  =                                     endif
2498  =                             ;
2499  =                                     if      not loader
2500  =                                     endif
2501  =                             to_dirtouser:
2502  =0F10 E90AFB      0A1D        jmp     dirtouser           ;copy directory entry to user
2503  =                             ;                           ;ret ;jmp goback
2504  =
2505  =                             ;
2506  =                             func19:
2507  =                             ;
2508  =                                     ;delete a file
2509  =                                     if      loader
2510  =0F13 C3                              ret
2511  =                                     endif
2512  =                             ;
2513  =                                     if      not loader
2514  =                                     endif
2515  =                             ;
2516  =                             ;
2517  =                             func20:
2518  =                             ;
2519  =                                     ;read a file
2520  =                             ;
2521  =0F14 E8C8FE      0DE2        CALL    parsave33r
2522  =0F17 E81AFF      0E34        call    check_fcb
2523  =0F1A E950FE      0D6D        jmp     sqread              ;jmp goback
2524  =
2525  =                             ;
2526  =                             func21:
2527  =                                     if      loader
2528  =0F1D C3                              ret
2529  =                                     endif
2530  =                             ;
2531  =                             ;
2532  =                                     if      not loader
2533  =                                     endif
2534  =                             ;
2535  =                             ;
2536  =                             func22:
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2537
2538  =                     ;
2539  =                           ;make a file
2540  =                           if      loader
2541  =0F1E C3                    ret
2542  =                           endif
2543  =                     ;
2544  =                           if      not loader
2545  =                           endif
2546  =                     ;
2547  =                     func23:
2548  =                     ;
2549  =                           ;rename a file
2550  =                           if      loader
2551  =0F1F C3                    ret
2552  =                           endif
2553  =                     ;
2554  =                           if      not loader
2555  =                           endif
2556  =                                                   ;ret ;jmp goback
2557  =                     ;
2558  =                     func24:
2559  =                     ;
2560  =                           ;return the login vector
2561  =                     ;
2562  =0F20 8B1E2210              MOV     BX,dlog
2563  =0F24 EB1B        0F41      jmps    sthlret
2564  =                     ;     ret ;jmp goback
2565  =                     ;
2566  =                     func25:
2567  =                     ;
2568  =                           ;return selected disk number
2569  =                     ;
2570  =0F26 A0A411                MOV     AL,curdsk
2571  =                     to_staret:
2572  =0F29 E95CF8        0788    jmp     staret
2573  =                     ;     ret ;jmp goback
2574  =                     ;
2575  =                     func26:
2576  =                     ;
2577  =                           ;set the subsequent dma address to info
2578  =                     ;
2579  =                                                   ;was MOV BX,INFO
2580  =0F2C 8916A011              MOV     dmaad,dx        ;dmaad = info
2581  =0F30 E9D1FA        0A04    JMP     setdata         ;to data dma address
2582  =                                                   ;ret ;jmp goback
2583  =                     ;
2584  =                     func27:
2585  =                     ;
2586  =                           ;return the login vector address
2587  =                     ;
2588  =0F33 881E3010              MOV     BX,alloca
2589  =0F37 EB08        0F41      jmps    sthlret
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2590
2591  =                     ;       ret ;jmp goback
2592  =                     ;
2593  =  094C               func28  EQU     setro
2594  =                     ;
2595  =                             ;write protect current disk
2596  =                     ;
2597  =                                                     ;ret ;jmp goback
2598  =                     ;
2599  =                     func29:
2600  =                     ;
2601  =                             ;return r/o bit vector
2602  =                     ;
2603  =0F39 8B1E2010                 MOV     BX,rodsk
2604  =0F3D EB02        0F41        jmps    sthlret
2605  =                     ;       ret ;jmp goback
2606  =                     ;
2607  =                     func30:
2608  =                     ;
2609  =                             ;set file indicators
2610  =                             if      loader
2611  =0F3F C3                      ret
2612  =                             endif
2613  =                     ;
2614  =                             if      not loader
2615  =                             endif
2616  =                                                     ;ret ;jmp goback
2617  =                     ;
2618  =                     func31:
2619  =                     ;
2620  =                             ;return address of disk parameter block
2621  =                     ;
2622  =                             if      loader
2623  =0F40 C3                      ret
2624  =                             endif
2625  =                     ;
2626  =                             if      not loader
2627  =                             endif
2628  =                     sthlret:
2629  =0F41 891E9B11                MOV     aret,bx
2630  =0F45 C3                      RET                             ;jmp goback
2631  =                     ;
2632  =                     func32:
2633  =                     ;
2634  =                             ;set user code
2635  =                             if      loader
2636  =                     ;
2637  =0F46 C3                      ret
2638  =                             endif
2639  =                             if      not loader
2640  =                             endif
2641  =                     ;
2642  =                     ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2643
2644  =
2645  =
2646  =
2647  =
2648  =0F47 C3
2649  =
2650  =
2651  =
2652  =
2653  =
2654  =
2655  =
2656  =
2657  =
2658  =
2659  =0F48 C3
2660  =
2661  =
2662  =
2663  =
2664  =
2665  =
2666  =
2667  =
2668  =
2669  =
2670  =
2671  =0F49 C3
2672  =
2673  =
2674  =
2675  =
2676  =
2677  =
2678  =
2679  =
2680  =
2681  =
2682  =
2683  =0F4A C3
2684  =
2685  =
2686  =
2687  =
2688  =
2689  =
2690  =
2691  =
2692  =0F4B C3
2693  =
2694  =
2695  =
```

```
func33:
;
        ;random disk read operation
        if      loader
        ret
        endif
;
        if      not loader
        endif
                                        ;ret ;jmp goback
;
func34:
;
        ;random disk write operation
        if      loader
        ret
        endif
;
;
        if      not loader
        endif
;
;
func35:
;
        ;return file size (0-65536)
        if      loader
        ret
        endif
;
        if      not loader
        endif
;
;
func36:
;
        ;set random record
;
        if      loader
        ret
        endif
        if      not loader
        endif
;
;
func37:
;
        if      loader
        ret
        endif
;
        if      not loader
```

COPYRIGHT © 1981 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2696
2697  =                             endif
2698  =                     ;
2699  =                     ;
2700  =                     goback:
2701  =                     ;
2702  =                             ;arrive here at end of processing to return to user
2703  =                     ;
2704  =0F4C 803E4B1000              cmp     resel,0
2705  =0F51 7424            0F77    JZ      retmon
2706  =                                                     ;reselection may have
2707  =                                                     ;taken place
2708  =0F53 8B1E9911                MOV     BX,info
2709  =0F57 803E611060              cmp     highext,zero_readro     ;user zero and read/only
2710  =0F5C 7504            0F62    jnz     goback1
2711  =0F5E 804F0880                or      f8[bx],interfc_bit      ;set interface attribute r/o
2712  =                     goback1:
2713  =0F62 C60700                  MOV     b[bx],0                 ;fcb(0)=0
2714  =0F65 A04D10                  MOV     AL,fcbdsk
2715  =0F68 0AC0                    OR      AL,AL
2716  =0F6A 740B            0F77    JZ      retmon
2717  =                                                     ;restore disk number
2718  =0F6C 8807                    MOV     [bx],al                 ;fcb(0)=fcbdsk
2719  =0F6E A04C10                  MOV     AL,olddsk
2720  =0F71 A24810                  MOV     linfo,AL
2721  =0F74 E863FE          0DDA    CALL    curselect
2722  =                     ;
2723  =                     ;       return from the disk monitor
2724  =                     retmon:
2725  =0F77 803E5F10FF              cmp     parcopfl,true           ;test if parameter block
2726  =0F7C 7503            0F81    jnz     retm1                   ;to be copied back to user
2727  =0F7E E890F5          0511    call    parunsave
2728  =0F81 A17B11          retm1:  mov     ax,aret
2729  =0F84 8BD8                    mov     bx,ax
2730  =                     retmonret:
2731  =0F86 C3                      RET
2732  =  078B               func38  EQU     funcret
2733  =  078B               func39  EQU     funcret
2734  =                     func40:
2735  =                             if      loader
2736  =0F87 C3                      ret
2737  =                             endif
2738  =                     ;
2739  =                             if      not loader
2740  =                             endif
2741  =                     ;
2742  =                     func47:
2743  =                     ;       chain to program
2744  =                     ;       move 128 byte from user dma to CCP's command buffer
2745  =                             if      loader
2746  =0F88 C3                      ret
2747  =                             endif
2748  =                     ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2749
2750  =                         if      not loader
2751  =                         endif
2752  =                 ;
2753  =                 ;
2754  =                 func48:
2755  =0F89 C3                  ret
2756  =                 ;
2757  =                 ;
2758  =                 func49:
2759  =                         if      loader
2760  =0FBA C3                  ret
2761  =                         endif
2762  =                         if      not loader
2763  =                         endif
2764  =                 ;
2765  =                 ;
2766  =                 ;******************** end BDOS filesystem part 3 ****************
2767                    end
```

CP/M 86 V.1.1 - vol T
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81-162

```
2768
2769  =                 eject ! include bdosdat.a86     ; data area
2770  =                 ;********************* BDOS data area **********************
2771  =                 ;
2772  =                 ;
2773  =  0000          dseg    cpmsegment
2774  =                 ;       Variables in data segment:
2775  =                         org     bdosdataseg
2776  =                 ;
2777  =                 ;     the follo ng pointer to the BIOS is used by the BDOS
2778  =                 ;        and changed by the NET when present in memory
2779  =                 ;
2780  =1000 A507        biosco  dw      offset bioscl   ;offset of 'call BIOS' instruction
2781  =1002 0000        bioscs  dw      00h             ;segment location of BIOS
2782  =  1000           biosptr equ     biosco
2783  =                 ;
2784  =                 ;
2785  =1004 A404        pererr  dw      persub          ;permanent error subroutine
2786  =1006 B004        seler   dw      selsub          ;select error subroutine
2787  =1008 B504        roderr  dw      rodsub          ;ro disk error subroutine
2788  =100A BA04        rofer   dw      rofsub          ;rofile error subroutine
2789  =                 ;
2790  =                 ;
2791  =                 ;
2792  =                 ;
2793  =                         if      not loader
2794  =                         endif
2795  =                 ;
2796  =                 ;
2797  =                                                 ;returned in bx
2798  =100C 0000        pag0    dw      0               ;address of user's page zero
2799  =100E 00          ip0     db      0               ;initial page value for IP register
2800  =                 ;
2801  =                 ;
2802  =                 ;
2803  =                 ;
2804  =                 ;     memory control block
2805  =                 ;
2806  =100F 0000        umembase        dw      0       ;user'sbase for memory request
2807  =1011 0000        umemlg  dw      0               ;length of memory req
2808  =1013 00          contf   db      0               ;flag indicates added memory is avail
2809  =                 ;
2810  =                 ;
2811  =1014 0000        hold_info       dw      0       ;save info
2812  =1016 0000        hold_spsave     dw      0       ;save user SP during program load
2813  =1018 0000        hold_sssave     dw      0       ;save user SS during program load
2814  =
2815  =101A 00          mod8080         db      0
2816  =                 ;
2817  =                 ;
2818  =                 ;       byte I/O variables:
2819  =                 ;
2820  =101B 00          compcol db      0               ;true if computing column position
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #

```
2821
2822  =101C 00              strtcol db      0               ;starting column position after read
2823  =101D 00              column  db      0               ;column position
2824  =101E 00              kbchar  db      0               ;initial key char = 00
2825  =                     ;
2826  =                     ;       common values shared between bdosi and bdos
2827  =                     ;
2828  =                     ;       data areas for file system:
2829  =                     ;
2830  =                     ;       initialized data
2831  =                     ;
2832  =101F E5              efcb    db      empty           ;0e5=available dir entry
2833  =1020 0000            rodsk   dw      0               ;read only disk vector
2834  =1022 0000            dlog    dw      0               ;logged-in disks
2835  =                     ;
2836  =                     ;       curtrka - alloca are set upon disk select
2837  =                     ;       (data must be adjacent, do not insert variables)
2838  =                     ;       (address of translate vector, not used)
2839  =                     ;
2840  =1024 0000            cdrmaxa dw      0               ;pointer to cur dir max value
2841  =1026 0000            curtrka dw      0               ;current track address
2842  =1028 0000            curreca dw      0               ;current record address
2843  =102A 0000            buffa   dw      0               ;pointer to directory dma address
2844  =102C 0000            dpbaddr dw      0               ;current disk parameter block address
2845  =102E 0000            checka  dw      0               ;current checksum vector address
2846  =1030 0000            alloca  dw      0               ;current allocation vector address
2847  =  0008               addlist equ     8               ;'$-buffa' = address list size
2848  =                     ;
2849  =                     ;       sectpt - offset obtained from disk parm block at dpbaddr
2850  =                     ;       (data must be adjacent, do not insert variables)
2851  =1032 0000            sectpt  dw      0               ;sectors per track
2852  =1034 00              blkshf  db      0               ;block shift factor
2853  =1035 00              blkmsk  db      0               ;block mask
2854  =1036 00              extmsk  db      0               ;extent mask
2855  =1037 0000            maxall  dw      0               ;maximum allocation number
2856  =1039 0000            dirmax  dw      0               ;largest directory number
2857  =103B 0000            dirblk  dw      0               ;reserved allocation bits for direct.
2858  =103D 0000            chksiz  dw      0               ;size of checksum vector
2859  =103F 0000            offsetv dw      0               ;offset tracks at beginning
2860  =1041                 endlist rs      0               ;end of list
2861  =  000F               dpblist equ     (offset endlist)-(offset sectpt)        ;size
2862  =                     ;
2863  =                     ;       local variables
2864  =1041 0000            tranv   dw      0               ;address of translate vector
2865  =1043 00              fcbcopied db    0               ;set true if copy$fcb called
2866  =1044 00              rmf     db      0               ;read mode flag for open$reel
2867  =1045 00              wflag   db      0               ;write flag sent to BIOS for blocking
2868  =1046 00              dirloc  db      0               ;directory flag in rename, etc.
2869  =1047 00              seqio   db      0               ;1 if sequential i/o
2870  =1048 00              linfo   db      0               ;low(info)
2871  =1049 00              dminx   db      0               ;local for diskwrite
2872  =104A 00              single  db      0               ;set true if single byte
2873  =                                                     ;allocation map
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. Box 579
PACIFIC GROVE, CA 93950

```
2874
2875  =104B 00              resel   db      0               ;reselection flag
2876  =104C 00              olddsk  db      0               ;disk on entry to bdos
2877  =104D 00              fcbdsk  db      0               ;disk named in fcb
2878  =104E 00              rcount  db      0               ;record count in current fcb
2879  =104F 00              extval  db      0               ;extent number and extmsk
2880  =1050 0000            vrecord dw      0               ;current virtual record
2881  =  1050               lvrecord equ    byte ptr vrecord ;low (vrecord)
2882  =1052 0000            arecord dw      0               ;current actual record
2883  =1054 0000            ablock  dw      0               ;current actual block# * blm sk
2884  =                     ;
2885  =                     ;       local variables for directory access
2886  =1056 00              dptr    db      0               ;directory pointer 0,1,2,3
2887  =1057 0000            drec    dw      0               ;directory record 0,1,...,dirmax/4
2888  =                     ;
2889  =                     ;       Special 8086 variables:
2890  =                     ;
2891  =1059 0000            srcha   dw      0               ;search address
2892  =105B 0000            user_parm_seg   dw      0       ;holds user parameter seg during load
2893  =105D 0000            retrn_es        dw      0       ;ES register returned to user
2894  =105F 00              parcopfl        db      0       ;true if parameter block copied
2895  =1060 00              parlg           db      0       ;length of parameter block
2896  =                             if      not loader
2897  =                             endif
2898  =                     ;
2899  =                     ;   added for user zero search
2900  =1061 00              high_ext        db      0       ;fcb high extend bits
2901  =1062 00              search_user0    db      0       ;search user 0 on 'open'
2902  =1063 0000            xdcnt   dw      0               ;dcnt saved on user 0 search
2903  =  1064               found_in_zero   equ     xdcnt+1 ;found file in user zero
2904  =1065 00              user_zero_pass  db      0       ;search user zero flag
2905  =                     ;
2906  =                     ;
2907  =                     ;
2908  =                     ;       error messages
2909  =                     ;
2910  =1066 42646F732045    dskmsg  db      'Bdos Err On '
2911        7272204F6E20
2912  =1072 203A2024        dskerr  db      ' : $'          ;filled in by errflg
2913  =1076 426164205365    permsg  db      'Bad Sector$'
2914        63746F7224
2915  =1081 53656C656374    selmsg  db      'Select$'
2916        24
2917  =1088 46696C6520      rofmsg  db      'File '
2918  =108D 522F4F24        rodmsg  db      'R/O$'
2919  =                     ;
2920  =                     ;       Local buffer area:
2921  =                     ;
2922  =1091                 loc_par_area    rb      40      ;local user parameter (FCB,...)
2923  =1089                 stackarea       rw      ssize   ;stack size
2924  =1159                 endstack        rb      0       ;top of stack
2925  =                     ;
2926  =                     ;********************* end BDOS data area *********************
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2927
2928 =                  ;
2929 =                  ;
2930 =                  ;   SYSTEM CONTROL BLOCK....Address returned by function 49
2931 =                  ;
2932 =                          org     biosoffset-6ch
2933 =1194 0000         SS_save         dw      0
2934 =1196 0000         SP_save         dw      0
2935 =1198 00           stack_flag      db      0h
2936 =1199 0000         info    dw      0                       ;information address
2937 =119B 0000         aret    dw      0                       ;address value to return
2938 =  119B            lret    EQU     byte ptr aret           ;low(aret)
2939 =                  ;
2940 =119D 0000         parametersegment dw     0       ;user parameter segment
2941 =                  ;
2942 =                          org     biosoffset-60h
2943 =                  ;
2944 =11A0 0000         dmaad   dw      0               ;user DMA address
2945 =11A2 0000         dmabase dw      0               ;user DMA base
2946 =11A4 00           curdsk  db      0               ;current user disk
2947 =11A5 00           usrcode db      0               ;current user number
2948 =11A6 0000         dcnt    dw      0               ;directory index
2949 =  11A6            ldcnt   equ     byte ptr dcnt   ;low(dcnt)
2950 =11A8 00           srchl   db      0               ;search length
2951 =11A9 0000         infosave        dw      0       ;save for FCB address
2952 =11AB 0000         srcha_seg       dw      0       ;DS segment from last search
2953 =11AD 0000         cbuffa          dw      0       ;address of directory buffer
2954 =                  ;
2955 =                  ;
2956 =11AF 00           background      db      0       ;background despool checks for abort
2957 =                                                  ;01h...despool in memory
2958 =                                                  ;03h....abort despool
2959 =11B0 00           desplchars      db      0       ;number of characters for despool
2960 =11B1 00           mxdos           db      0       ;BDOS diskio flag
2961 =11B2 00           subflag db      0               ;submit flag
2962 =11B3 00           buf_flg db      0               ;xsub flag if buffer not empty
2963 =11B4 0000         ercode  dw      0               ;error return code for submit
2964 =                  ;
2965 =11B6 00           listcp  db      0               ;listing toggle...set by ctl-p
2966 =11B7 00           curdrvs db      0               ;current drives to search
2967 =11B8 00                   db      0
2968 =11B9 00                   db      0
2969 =11BA 00           intflag db      0
2970 =11BB 0000         roldate dw      0               ;rolander date
2971 =11BD              roltime rs      3               ;rolander time
2972 =11C0              asdate  rs      8               ;ascii date
2973 =11C8 00                   db      0               ;end string
2974 =11C9              astime  rs      9               ;ascii time
2975 =11D2                      rs      14
2976 =11E0 50           colwdth db      80              ;screen column width
2977 =11E1 00                   db      0               ;end string
2978 =11E2                      rs      24              ;expansion space
2979 =11FA 00                   db      0               ;end of this data area
```

```
2980
2981  =                    ;
2982  =                    ;
2983                            end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2984
2985                        eject
2986                        ;************************ end of BDOS ***********************
2987                        end
2988
2989
2990  END OF ASSEMBLY.  NUMBER OF ERRORS:  0.  USE FACTOR: 39%
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
ABLOCK          1054 V  1282  2883#
ADDH            0988 L  1410#
ADDLIST         0008 N  1127  2847#
ALLMODULES      0F05 L  2481  2483#
ALLOCA          1030 V  1686  1696  2588  2846#
ALLOCT          08B4 L  1271#  2260
ARECORD         1052 V  1162  1186  1195  1268  1273  1280  1286  2882#
ARET            119B V   321  2629  2728  2937#  2938
ASDATE          11C0 V  2972#
ASTIME          11C9 V  2974#
ATRAN           08BB L  1277#  2264
B               0000 N   55#  510   612   616   626   633   812   1138  1143  1430
                        1690  1716  1873  1894  2178  2352  2360  2429  2476  2480
                        2713
BACKGROUND      11AF V  2956#
BACKSP          06FD L   865#  869
BACKSP1         0706 L   859   870#
BACKUP          0609 L   667#  813   867
BACKX           06B1 L   805#  814
BDOSCODESIZE    0C00 N    28#   29
BDOSDATASEG     1000 N    29#  2775
BDOSE           0406 L   256#
BDOSE0          0434 L   276   281#
BDOSE1          0447 L   287   289#
BDOSINT         00E0 N   170#
BDOSOFFSET      0400 N    26#   29   192   246
BIOS            1200 L    79#   80    81    82    83    85    86    87    88    89
                          90    91    92    93    94    95    96    97    98    99
                         100   101
BIOSCL          07A5 L  1034#  2780
BIOSCO          1000 V  2780#  2782
BIOSCS          1002 V  2377  2380  2781#
BIOSOFFSET      1200 N    27#   78  2932  2942
BIOSPTR         1000 V  2782#
BLKMSK          1035 V  1284  2853#
BLKSHF          1034 V  1217  1279  2852#
BOOTF           1200 L    80#
BRANCH          0497 L   329   332   335#
BUFFA           102A V  1123  1126  1340  1408  1535  1546  2843#
BUFFLG          1183 V  2962#
BUFFNZERO       0CC3 L  2067  2073#
CALLOPEN        0E7C L  2403  2407#
CBUFFA          11AD V  1124  2953#
CCPENT          0000 L    68#
CDRMAXA         1024 V  1113  1376  1447  1702  2840#
CHECKA          102E V  1488  2845#
CHECKFCB        0E34 L  2356#  2522
CHECKSUM        09C3 L  1475#  1607
CHECKWRITE      0976 L  1396#
CHKSIZ          103D V  1480  2858#
CKRODIR         0963 L  1380#
CKROFILE        0966 L  1384#
CLOSE           0C77 L  2023#  2151  2463
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
CLRHODNUM       0997 L  1427#  2401  2482
CMPECO          092C L  1342#  1345
CMPECS          0923 L  1337#  1486
COLUMN          101D V   609   661   638   724   758   807   862  2823#
COLWDTH         11E0 V  2976#
COMPCDR         09A6 L  1444#  1455  1497  1808
COMPCOL         101B V   586   759   857   863   868  2820#
COMPEXT         0B11 L  1747#  1841
COMPOUT         05BE L   588   605#
CONBO           059B L   55&   574#
CONB1           059E L   546   577#
CONDRK          0579 L   543#   593   990
CONECH          0556 L   517#   916
CONIN           0547 L   382   506#  519   739
CONINF          1209 L    83#  1011
CONOUF          120C L    85  1020
CONOUT          05A1 L   582#   645   654   660   683   692   698   700   911
CONSTF          1206 L    82#  1016
CONTF           1013 V  2808#
COPYDIR         08F3 L  1915#  1953
COPYFCB         0BEF L  1909#
CPMSEGMENT      0000 N    24#    63   191  2773
CPYDIRLOC       0B0B L  1740#
CR              000D N   116#   532   697   743   908
CRLF            062E L   374    684   695#  788
CRLFP           0616 L   679#   822   834
CRLFP0          061E L   687#   693
CTL             005E N   118#   644
CTLC            0003 N   105#   347   559   888
CTLCCHECK       0598 L   560    566#  895
CTLCCHECK1      0598 L   570#
CTLE            0005 N   106#   732
CTLH            0008 N   107#   538   623   675   749
CTLOUT          05E6 L   636#   849   884
CTLP            0010 N   108#   794
CTLR            0012 N   109#   827
CTLS            0013 N   110#   555
CTLU            0015 N   111#   819
CTLX            0018 N   112#   801
CTLZ            001A N   113#
CURDRVS         11B7 V  2385  2966#
CURDSK          11A4 V   375  1103  1351  1362  2284  2308  2341  2386  2570  2946#
CURRECA         1028 V  1117  2842#
CURSELECT       0DDA L  2306#  2347  2391  2721
CURTRKA         1026 V  1115  1155  1200  2841#
DCNT            11A6 V  1183  1446  1559  1571  1578  1580  1896  2027  2425  2948#
                        2949
DESPLCHARS      11B0 V  2959#
DIOCOMP         0832 L  1164  1175#
DIRBLK          103B V  1695  2857#
DIRINP          0758 L   942   947#
DIRLOC          1046 V  1743  1768  1861  2868#
DIRMAX          1039 V  1374  1577  2856#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #_____

```
DIRREC          0004 N   132#  134
DIRTOUSER       0A1D L  1543#  2502
DISKEOF         0DAF L  2244   2252  2261  2269#
DISKF           000C N   202#
DISKIO          000D N    72#   266
DISKREAD        0D72 L  2230#
DLOG            1022 V  2283   2300  2302  2384  2562  2834#
DMAAD           11A0 V   239   1528  1547  2580  2944#
DMABASE         11A2 V   471   1526  2945#
DM1NX           1049 V  2871#
DMPOSITION      0373 L  1215#  1264
DMSEL           0CDF L  2076   2093#
DPBADDR         102C V  1131   2844#
DPBLIST         000F N  1133   2861#
DPTR            1056 V  1409   1600  2836#
DREC            1057    1187   1479  1487  2887#
DSKERR          1072 V   377   2912#
DSKMAP          0010 N   147#  1248  1631  1633  2040  2048
DSKMSG          1066 V   378   2910#
DSKMSK          0003 N   134#  1593  1863
DSKSHF          0002 N   133#  1134
DVERS           0022 N    40#   234
ECHOC           0566 L   520    529#  639
EFCB            101F V  2832#
EMPTY           00E5 N   128#  1716  1803  2832
ENDDIR          FFFF N   124#  1570
ENDLIST         1041 V  2860#  2861
ENDMERGE        0D06 L  2115   2130#
ENDOFDIR        0A31 L  1556#  1711  1790  1947  1961  2037  2154  2189  2440
ENDSEARCH       0BA6 L  1818   1859#
ENDSTACK        1159 V   279   2924#
EOFRE           0669 L   745#   748
ERCODE          1184 V  2963#
ERRFLG          04C3 L   346    366   370#
EXTMSK          1036 V  1315   1751  2166  2854#
EXTNUM          000C N   138#  1293  1829  1944  1952  1981  2103  2120  2158  2177
EXTVAL          104F V  1225   1317  2879#
FOFREE          0533 L   368    464#  571
FS              0008 N   158#  2323  2327  2711
FALSE           0000 N    44#   312  1143  1786  1788  1849  2417
FCBCOPIED       1043 V  2131   2149  2169  2865#
FCBDSK          104D V   324   2344  2714  2877#
FCBLEN          0020 N   131#   132   148   149  1633  1912  2048  2103
FCBNZERO        0CB9 L  2062   2065#
FCBSHF          0005 N   135#  1595
FOUNDINZERO     1064 V  2404   2418  2903#
FUNC0           0531 L  194    457#
FUNC1           073F L   196   913#
FUNC10          077E L   200   981#
FUNC11          0785 L   200   987#
FUNC12          03EC L   203   233#
FUNC13          0E43 L   203  2373#
FUNC13A         0E4F L  2379  2381#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
FUNC14          0DDA L   203  2391#
FUNC15          0E64 L   203  2396#
FUNC16          0EDE L   205  2458#
FUNC17          0EE4 L   205  2466#
FUNC18          0F0F L   205  2492#
FUNC19          0F13 L   205  2506#
FUNC2           05F7 L   196   919#
FUNC20          0F14 L   207  2517#
FUNC21          0F1D L   207  2526#
FUNC22          0F1E L   207  2536#
FUNC23          0F1F L   207  2547#
FUNC24          0F20 L   209  2558#
FUNC25          0F26 L   209  2566#
FUNC26          0F2C L   209  2575#
FUNC27          0F33 L   209  2584#
FUNC28          094C L   211  2593#
FUNC29          0F39 L   211  2599#
FUNC3           0744 L   196   923#
FUNC30          0F3F L   211  2607#
FUNC31          0F40 L   211  2618#
FUNC32          0F46 L   213  2632#
FUNC33          0F47 L   213  2644#
FUNC34          0F48 L   213  2655#
FUNC35          0F49 L   213  2667#
FUNC36          0F4A L   215  2678#
FUNC37          0F4B L   215  2689#
FUNC38          078B L   215  2732#
FUNC39          078B L   215  2733#
FUNC40          0F87 L   217  2734#
FUNC47          0F88 L   218  2742#
FUNC48          0F89 L   219  2754#
FUNC49          03F1 L   220   238#
FUNC50          0532 L   221   461#
FUNC51          053B L   221   468#
FUNC52          0545 L   221   477#
FUNC53          0546 L   221   488#
FUNC54          0546 L   223   489#
FUNC55          0546 L   223   490#
FUNC56          0546 L   223   491#
FUNC57          0546 L   223   492#
FUNC58          0546 L   225   493#
FUNC59          0530 L   225   448#
FUNC6           0749 L   198   937#
FUNC7           0767 L   198   957#
FUNC8           076C L   200   963#
FUNC9           0772 L   200   970#
FUNCRET         078B L   995# 2732  2733
FUNCTAB         0380 V   196#  340
FWFMSK          0080 N   143# 1437  2033
GETDM           088B L  1244# 1267
GETDMD          08A0 L  1251  1258#
GETDPTRA        0981 L  1382  1405# 1630  1715  1813  1922  1971  2041
GETEXTA         08D7 L  1290# 1314  1967  2479
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, A 93950
SER. #

```
GETFCB          08EE L  1305#  2205  2235
GETIOBF         1239 L   100#   960
GETMODNUM       098D L  1418#  1429  1434  2032
GETSEGTABF      1236 L    99#
GOBACK          0F4C L   326  2700#
GOBACK1         0F62 L  2710  2712#
GOERR           07AB L  1076#  1179  1402
GTFCBA          08E1 L  1297#  1308  1322  2246
HIGHEXT         1061 V  2322  2328  2357  2426  2443  2450  2709  2900#
HOLDINFO        1014 V  2811#
HOLDSPSAVE      1016 V  2812#
HOLDSSSAVE      1018 V  2813#
HOME            0816 L  1150#  1699  1777
HOMEF           1218 L    89#  1152
INBDOS          0419 L   264   268#
INDEX           08A5 L  1262#  2255
INDICO          0C0F L  1946#  1955
INDICATORS      0C0A L  1942#
INFO            1199 V   318   395   398   433   726  1246  1292  1300  1422  1765
                        1920  1973  2006  2045  2157  2321  2331  2351  2359  2428
                        2446  2475  2708  2936#
INFOSAVE        11A9 V   396   432  2951#
INITO           0ABE L  1689#  1692
INIT2           0ADC L  1708#  1717  1738
INITCS          09EE L  1491   1501#
INITIALIZE      0AAF L  1677#  2303
INTERFCBIT      0080 N   159#  2323  2711
INTERRUPTBIT    0200 N   171#   286
INTFLAG         11BA V  2969#
INVIS           000A N   154#
IPO             100E V  2799#
JMPFOFREE       04C0 L   349   367#
JMPGOERR        097E L  1392  1401#
JMPSRCHFIN      0B4C L  1793#  1810
KBCHAR          101E V   508   544   576  2824#
L11             05BD L   601   603#
L19             066C L   744   746#
L23             072B L   892   894#
L24             0732 L   901   903#
L26             0762 L   950   952#
L41             0A87 L  1638  1640#
L42             0AAA L  1665  1668#
L67             0DC8 L  2291  2293#
LDCNT           11A6 V  1592  1862  2949#
LDROFFSET       0000 N    25#    67
LF              000A N   117#   534   630   699   747
LINELEN         06D3 L   762   829#
LINFO           1048 V   319  2307  2336  2720  2870#
LISTCP          11B6 V   600   797  2965#
LISTF           120F L    86#  1007
LISTFILE        0005 N    71#   263
LISTSTF         122D L    96#
LOADER          FFFF N    47#   186   485   487   567  1023  1025  1611  1615  1666
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #____ ____ ____

```
                    1901  1904  1932  1935  2141  2196  2274  2495  2499  2509
                    2513  2527  2532  2540  2544  2550  2554  2610  2614  2622
                    2626  2635  2639  2647  2651  2658  2663  2670  2674  2682
                    2685  2691  2695  2735  2739  2745  2750  2759  2762  2793
                    2896
LOCPARAREA      1091 V   397   431  2922#
LRET            119B V   994  1731  1864  2026  2137  2251  2938#
LSTFC8          001F N   143#
LSTREC          007F N   129#
LVRECORD        1050 V  1219  1283  1310  1329  2236  2831#
MAXALL          1037 V  1136  1663  1681  2855#
MAXEXT          001F N   139# 1758  2161
MAXMOD          000F N   142# 2182
MEMBASE         0000 N   164#
MEMDESCRLG      0004 N   166#
MEMLG           0002 N   165#
MERGEO          0CA7 L  2049# 2098
MERGED          0CC7 L  2051  2077#
MERGERR         0D0E L  2075  2088  2135#
MERGEZERO       0C3C L  2012# 2079  2081
MOD8080         101A V  2815#
MODNUM          000E N   141# 1423  2177
MOVE            07AD L  1084# 1128  1134  1551  1924  1976
MXDOS           11B1 V   269   312  2960#
NAMLEN          000F N   145# 1959  2035  2187  2430  2486
NBIOSFUNCS      0015 N   102#
NEWCHECKSUM     09C1 L  1472# 1509
NFUNCS          0036 N   229#  331
NFUNCS80        0029 N   227#  328   330
NOSELECT        0E2A L  2338  2348#
NOTBACKSP       05DE L   624   628#
NOTC            072B L   890   898#
NOTE            06A2 L   783   792#
NOTFOUND        00FE N   157# 2404  2418
NOTR            0682 L   750   763#
NOTINTERFC8IT   007F N   160# 2327
NOTP            06AC L   795   799#
NOTR            0708 L   828   873#
NOTRUB          0692 L   767   780#
NOTU            06CF L   820   825#
NOTX            06C4 L   802   815#
NOWRITE         0940 L  1358# 1398  2028
NXTREC          0020 N   149#  150  1301  1974
OFF             0000 N    50#   51
OFFSETV         103F V  1203  2859#
OLDBDOS         046F L   301   317#
OLDDSK          104C V  2342  2719  2876#
ON              FFFF N    49#
OPEN            0C20 L  1957# 2408
OPEN1           0C28 L  1962# 2434
OPENCOPY        0C2A L  1965# 2202
OPENERR         0D67 L  2183  2194  2209#
OPENMOD         0D40 L  2163  2175#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
OPENRO           0D4B L  2171  2186#
OPENR1           0D5C L  2174  2190  2200#
OPENR2           0D5F L  2204#
OPENRCNT         0C60 L  2001  2003  2005#
OPENREEL         0D13 L  2145# 2245
OPENUSERZERO     0EA8 L  2427#
OPENX            0EBE L  2409  2435  2439#
OPENXO           0EDB L  2445  2449  2453#
PAGO             100C V  2798#
PARAMETERSEGMENT 119D V   260   313   399   430   975   983  1550  2471  2940#
PARCOPFL         105F V   322   392  2725  2894#
PARLG            1060 V   393   428  2895#
PARRET           0503 L   348   405#
PARSAVE          04DC L   384#  418
PARSAVE33        0504 L   407#  2312  2469
PARSAVE33R       0DE2 L  2311#  2400  2462  2521
PARSAVE36        0509 L   413#  2316
PARSAVE36R       0DE7 L  2315#
PARUNSAVE        0511 L   422#  2727
PCTLH            0611 L   669   673#
PDOLLAR          0801 L  1721  1728  1732#
PERERR           1004 V  1178  2785#
PERMSG           1076 V   345  2913#
PERSUB           04A4 L   344#  2785
PRINT            0638 L   379   381   703#  720   976
PSCONMON         050C L   411   417#
PUNCHF           1212 L    87#  198
QSELECT          0F0A L  2477  2487#
RANREC           0021 N   150#
RCOUNT           104E V  1313  1332  2237  2878#
RDBUFF           0822 L  1160#  1520  2266
RDDIR            0A46 L  1575#  1710  1789
RDDIRO           0A5A L  1584  1589#
RDDIRBUF         09FE L  1516#  1605
RDECH1           070E L   778   879#
RDECHO           0708 L   875#
READ             064D L   722#  810   824   934
READEN           0732 L   745   904#
READERF          1215 L    88#  927
READF            1227 L    94#  1163
READNO           0660 L   738#  791
READNX           065E L   734#  754   770   798   902
RECCNT           000F N   146#  1299  1301  1985  2007  2120
RECORDOK         0D9B L  2239  2253#
RECSIZ           0080 N   130#  132  1339  1548
REPO             06DB L   841#  852
REP1             06EE L   843   853#
RESEL            104B V   325  2330  2704  2875#
RESELECT         0DEA L  2313  2319#  2485
RESELECTO        0E02 L  2326  2329#
RESTORESTACK     0533 L   465#
RETO             0578 L   512   521   533   535   537   539   541#  551
RET1             05DB L   611   614   619   627#  631
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
RET10           0C09 L  1936# 1948  1963
RET11           0C6B L  2010# 2017
RET12           0C76 L  2021# 2029  2034  2038
RET13           0D12 L  2138# 2155
RET2            0608 L   664#  690   708
RET20           0E33 L  2296  2353#
RET30           0E39 L  2414# 2419  2441
RET4            0815 L  1110  1147# 1177
RET5            093C L  1390  1399  1415#
RET6            0??0 L  1451# 1456  1482
RET7            09F0 L  1494  1498  1504#
RET71           0A77 L  1602  1613#
RET8            0A3D L  1563  1566#
RET8B           0B21 L  1712  1760#
RET9            0BB7 L  1868#
RETM1           0F?1 L  2726  2728#
RETMON          0r/7 L   951  2705  2716  2724#
RETMONRET       0F86 L  2730#
RETRNES         105D V   298   306  2893#
RETSELECT       0811 L  1140  1144#
RMF             1044 V  2192  2231  2866#
RODERR          1008 V  1400  2787#
RODMSG          108D V   358  2918#
RODSK           1020 V  1361  1369  1371  2383  2603  2833#
RODSUB          04B5 L   357# 2737
ROFER           100A V  1391  2788#
ROFILE          0009 N   153# 1336
ROFMSG          1088 V   362  2917#
ROFSUB          04BA L   361# 2738
ROLDATE         11BB V  2970#
ROLTIME         118D V  2971#
RUBOUT          007F N   114#  610   766
SAVEDCNTPOS2    0BDB L  1845  1890#
SAVESTACKS      041E L   267   272#
SCAN3           0AAA L  1662  1670#
SCNDK           0A78 L  1624# 1736
SCNDK0          0A81 L  1634# 1675
SCNDK1          0A97 L  1644  1651#
SCNDK2          0A9E L  1650  1658#
SEARCH          0B34 L  1771#
SEARCHI         0B22 L  1763# 1775  2432
SEARCHI1        0B2A L  1767#
SEARCHN         0B3D L  1781# 1899  2433
SEARCHNJMP      0BEC L  1872  1874  1876  1892  1895  1898#
SEARCHUSER0     1062 V  1849  1875  2405  2410  2417  2901#
SECTPT          1032 V  1132  1197  2851# 2861
SECTRAN         1230 L    97# 1207
SEEK            084C L  1191# 2265
SEEKCOPY        0C03 L  1925# 2132
SEEKDIR         083C L  1181# 1604  1927
SELDSKF         121B L    90# 1106
SELECTDISK      07B9 L  1098# 2289
SELER           1006 V  1073  2786#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
SELERROR        07A8 L  1071# 2292
SELMSG          1081 V   354  2915#
SELSUB          04B0 L   353# 2786
SEQIO           1047 V  1323  2227  2869#
SETCDISK        0932 L  1348# 1370  2301
SETCDR          09B1 L  1453# 1737
SETDATA         0A04 L  1513  1524# 2581
SETDIR          0A10 L  1510  1518  1531#
SETDMA          0A18 L  1529  1537#
SETDMBF         1233 L    98#  473  1527  1534
SETDMF          1224 L    93# 1540
SETENDDIR       0A3E L  1568# 1587  1705  1776  1882
SETFC1          0915 L  1325  1327#
SETFCB          0909 L  1320# 2267
SETFWF          099E L  1433# 1979  2213
SETIOBF         123C L   101#  967
SETLRET1        078C L   999# 2212  2270
SETRO           094C L  1367# 1499  2593
SETSECF         1221 L    92# 1209
SETTRKF         121E L    91# 1204
SFUNC86         002F N   228#  330
SINGLE          104A V  1137  1250  1643  2050  2872#
SLECT           0DB2 L  2281# 2309  2388
SPSAVE          1196 V   274   282   311  2934#
SQREAD          0D6D L  2225# 2523
SRCH            0B34 L  1772# 1945  1960  2036  2188  2488
SRCHA           1059 V  1766  1800  2891#
SRCHASEG        11AB V  2472  2952#
SRCHEXT         0B87 L  1830  1836#
SRCHFIN         0BCF L  1794  1880#
SRCHL           11A8 V  1769  1814  2950#
SRCHLOOP        0B6A L  1816# 1858
SRCIN           0B3D L  1782# 1851  1954
SRCHN1          0B4F L  1791  1796#
SRCINEXT        0B61 L  1804  1812#
SRCHNM          0BB8 L  1834  1870#
SRCHOK          0B9E L  1822  1826  1835  1852# 1878
SSIZE           0050 N    60# 2923
SSSAVE          1194 V   273   284   310  2933#
STACKAREA       10B9 V  2923#
STACKFLAG       1198 V   275   466  2935#
STARET          0788 L   235   917   928   946   955   961   992# 1002  1744  1887
                        2207  2572
STHLRET         0F41 L   240  2563  2589  2604  2628#
STRTCOL         101C V   689   725   789   806  2822#
SUBDR           09BA L  1463# 1481  1583
SUBFLAG         11B2 V  2961#
TAB             0009 N   115#  536   653
TABO            05FC L   657#  663
TABOUT          05F7 L   525   640   651#  717   919
TESTF           0000 N    51#
THEBIOS         07A2 L  1008  1013  1017  1022#
TOCONINF        0795 L   514   554   558   954  1010#
```

COPYRIGHT © 1981, 1982
Digital Research
P.O. Box 579
PACIFIC GROVE, CA 93950
SER. #_____

```
TOCONOUF        079F L   596   671   676   944  1019#
TOCONSTF        079A L   549   945   948  1015#
TODIRTOUSER     0F10 L  2489  2501#
TOLISTF         0790 L   198   602  1006#
TOREADNO        06A0 L   790#  872
TOSTARET        0F29 L  2452  2571#
TRANV           1041 V  1121  1206  2864#
TRUE            FFFF N    43#   44    47   269   392  1138  1145  1474  1709  1844
                       1873  1875  1877  2131  2231  2330  2405  2410  2725
TRYZERO         0E8A L   413  2416#
UBYTES          000D N   140#  1825
UMEMBASE        100F V  2806#
UMEMLG          1011 V  2807#
USERPARMSEG     105B V  2892#
USERZEROPASS    1065 V  1786  1844  1877  2904#
USRCODE         11A5 V   719  2350  2402  2947#
VRECORD         1050 V  2248  2880#  2881
W               0000 N    56#  1078  1703  2016
WAITERR         04BD L   355   359   364#
WBOOTF          1203 L    81#   467
WFLAG           1045 V  1167  2867#
WRBUFF          082F L  1168#  1512
WRBUFFLG        082B L  1166#
WRDIR           09F1 L  1507#  1928
WRITEF          122A L    95#  1174
XCHECKFCB       0E42 L  2358  2361#
XDCNT           1063 V  1897  2421  2902#  2903
ZEROREADRO      0060 N   156#  2328  2357  2426  2709
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        title   '8086 Disk I/O Drivers'

                    ;*********************************************
                    ;*                                           *
                    ;* Basic Input/Output System (BIOS) for      *
                    ;* CP/M-86 Configured for iSBC 86/12 with    *
                    ;* the iSBC 204 Floppy Disk Controller       *
                    ;*                                           *
                    ;* (Note:  this file contains both embedded  *
                    ;* tabs and blanks to minimize the list file *
                    ;* width for printing purposes.  You may wish*
                    ;* to expand the blanks before performing    *
                    ;* major editing.)                           *
                    ;*********************************************

                    ;       Copyright (C) 1980,1981
                    ;       Digital Research, Inc.
                    ;       Box 579, Pacific Grove
                    ;       California, 93950
                    ;
                    ;       (Permission is hereby granted to use
                    ;       or abstract the following program in
                    ;       the implementation of CP/M, MP/M or
                    ;       CP/NET for the 8086 or 8088 Micro-
                    ;       processor)


FFFF                true            equ -1
0000                false           equ not true

                    ;*********************************************
                    ;*                                           *
                    ;* Loader_bios is true if assembling the     *
                    ;* LOADER BIOS, otherwise BIOS is for the    *
                    ;* CPM.SYS file.  Blc_list is true if we     *
                    ;* have a serial printer attached to BLC8538 *
                    ;* Bdos_int is interrupt used for earlier    *
                    ;* versions.                                 *
                    ;*                                           *
                    ;*********************************************

FFFF                loader_bios     equ true
FFFF                blc_list        equ true
00E0                bdos_int        equ 224 ;reserved BDOS Interrupt

                        IF      not loader_bios
                    ;-----------------------------------------------
                    ;;                                            ;
                    bios_code       equ 2500h                     ;
                    ccp_offset      equ 0000h                     ;
                    bdos_ofst       equ 0B06h ;BDOS entry point   ;
                    ;;                                            ;
```

cp/m 86 v.1.1 - vol I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER # C81-162

```
                        ;-----------------------------------------
                                ENDIF   ;not loader_bios
                        
                                IF      loader_bios
                        ;-----------------------------------------
                        ;;                                      :
0200                    bios_code       equ 1200h ;start of LDBIOS
0003                    ccp_offset      equ 0003h ;base of CPMLOADER
0406                    bdos_ofst       equ 0406h ;skipped BDOS entry
                        ;;                                      :
                        ;-----------------------------------------
                                ENDIF   ;loader_bios
                        
00DA                    csts            equ 0DAh  ;i8251 status port
00D8                    cdata           equ 0D8h  ;   "  data port
                        
                                IF      blc_list
                        ;-----------------------------------------
                        ;;                                      :
0041                    lsts            equ 41h ;2651 No. 0 on BLC8538 status port
0040                    ldata           equ 40h ;  "  "  "  "    "   data port
0060                    blc_reset       equ 60h ;reset selected USARTS on BLC8538
                        ;;                                      :
                        ;-----------------------------------------
                                ENDIF   ;blc_list
                        
                        ;***********************************************
                        ;*                                             *
                        ;*    Intel iSBC 204 Disk Controller Ports     *
                        ;*                                             *
                        ;***********************************************
                        
00A0                    base204         equ 0a0h        ;SBC204 assigned address
                        
00A0                    fdc_com         equ base204+0   ;8271 FDC out command
00A0                    fdc_stat        equ base204+0   ;8271 in status
00A1                    fdc_parm        equ base204+1   ;8271 out parameter
00A1                    fdc_rslt        equ base204+1   ;8271 in result
00A2                    fdc_rst         equ base204+2   ;8271 out reset
00A4                    dmac_adr        equ base204+4   ;8257 DMA base address out
00A5                    dmac_cont       equ base204+5   ;8257 out control
00A6                    dmac_scan       equ base204+6   ;8257 out scan control
00A7                    dmac_sadr       equ base204+7   ;8257 out scan address
00A8                    dmac_mode       equ base204+8   ;8257 out mode
00A8                    dmac_stat       equ base204+8   ;8257 in status
00A9                    fdc_sel         equ base204+9   ;FDC select port (not used)
00AA                    fdc_segment     equ base204+10  ;segment address register
00AF                    reset_204       equ base204+15  ;reset entire interface
                        
000A                    max_retries     equ 10          ;max retries on disk i/o
                                                        ;before perm error
000D                    cr              equ 0dh         ;carriage return
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                ;*********************************************
                                ;*                                           *
                                ;* BIOS Jump Vector for Individual Routines  *
                                ;*                                           *
                                ;*********************************************

1200 E93C00     123F  jmp INIT      ;Enter from BOOT ROM or LOADER
1203 E96100     1267  jmp WBOOT     ;Arrive here from BDOS call 0
1206 E96100     126A  jmp CONST     ;return console keyboard status
1209 E96700     1273  jmp CONIN     ;return console keyboard char
120C E96E00     127D  jmp CONOUT    ;write char to console device
120F E97600     1288  jmp LISTOUT   ;write character to list device
1212 E98800     129D  jmp PUNCH     ;write character to punch device
1215 E98500     129D  jmp READER    ;return char from reader device
1218 E9D000     12EB  jmp HOME      ;move to trk 00 on cur sel drive
121B E9AC00     12CA  jmp SELDSK    ;select disk for next rd/write
121E E9DF00     1300  jmp SETTRK    ;set track for next rd/write
1221 E9E100     1305  jmp SETSEC    ;set sector for next rd/write
1224 E9EA00     1311  jmp SETDMA    ;set offset for user buff (DMA)
1227 E9F500     131F  jmp READ      ;read a 128 byte sector
122A E9F600     1323  jmp WRITE     ;write a 128 byte sector
122D E96200     1272  jmp LISTST    ;return list status
1230 E9D700     130A  jmp SECTRAN   ;xlate logical->physical sector
1233 E9E000     1316  jmp SETDMAB   ;set seg base for buff (DMA)
1236 E9E200     131B  jmp GETSEGT   ;return offset of Mem Desc Table
1239 E96400     12A0  jmp GETIOBF   ;return I/O map byte (IOBYTE)
123C E96400     12A3  jmp SETIOBF   ;set I/O map byte (IOBYTE)

                                ;*********************************************
                                ;*                                           *
                                ;* INIT Entry Point, Differs for LDBIOS and  *
                                ;* BIOS, according to 'Loader_Bios' value    *
                                ;*                                           *
                                ;*********************************************

                                INIT:   ;print signon message and initialize hardware
123F 8CC8                               mov ax,cs       ;we entered with a JMPF so use
1241 8ED0                               mov ss,ax       ; CS: as the initial value of SS:,
1243 8ED8                               mov ds,ax       ;       DS:,
1245 8EC0                               mov es,ax       ;       and ES:
                                        ;use local stack during initialization
1247 BCA916                             mov sp,offset stkbase
124A FC                                 cld             ;set forward direction
                                
                                        IF      not loader_bios
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        ;----------------------------------------
                        ;|                                      |
                                ; This is a BIOS for the CPM.SYS file.
                                ; Setup all interrupt vectors in low
                                ; memory to address trap

                                push ds         ;save the DS register
                                mov ax,0
                                mov ds,ax
                                mov es,ax       ;set ES and DS to zero
                                ;setup interrupt 0 to address trap routine
                                mov int0_offset,offset int_trap
                                mov int0_segment,CS
                                mov di,4
                                mov si,0        ;then propagate
                                mov cx,510      ;trap vector to
                                rep movs ax,ax  ;all 256 interrupts
                                ;BDOS offset to proper interrupt
                                mov bdos_offset,bdos_ofst
                                pop ds          ;restore the DS register

                        ;**********************************************
                        ;*                                            *
                        ;* National "BLC 8538" Channel 0 for a serial*
                        ;* 9600 baud printer - this board uses 8 Sig-*
                        ;* netics 2651 Usarts which have on-chip baud*
                        ;* rate generators.                           *
                        ;*                                            *
                        ;**********************************************

                                mov al,0FFh
                                out blc_reset,al ;reset all usarts on 8538
                                mov al,4Eh
                                out ldata+2,al  ;set usart 0 in async 8 bit mode
                                mov al,3Eh
                                out ldata+2,al  ;set usart 0 to 9600 baud
                                mov al,37h
                                out ldata+3,al  ;enable Tx/Rx, and set up RTS,DTR
                        ;|                                      |
                        ;----------------------------------------
                                ENDIF   ;not loader_bios

                                IF      loader_bios
                        ;----------------------------------------
                        ;|                                      |
                                ;This is a BIOS for the LOADER
124B 1E                         push ds         ;save data segment
124C B80000                     mov ax,0
124F 8ED8                       mov ds,ax       ;point to segment zero
                                ;BDOS interrupt offset
1251 C70680030604               mov bdos_offset,bdos_ofst
1257 8C0E8203                   mov bdos_segment,CS ;bdos interrupt segment
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                ;!                                      !
                                ;---------------------------------------
                                        ENDIF   ;loader_bios

125C BB1514                             mov bx,offset signon
125F E85A00     12BC                    call pmsg       ;print signon message
1262 B100                               mov cl,0        ;default to dr A: on coldstart
1264 E99CED     0003                    jmp ccp         ;jump to cold start entry of CCP

1267 E99FED     0009 WBOOT:             jmp ccp+6       ;direct entry to CCP at command level

                                        IF      not loader_bios
                                ;---------------------------------------
                                ;!                                      !
                                int_trap:
                                        cli             ;block interrupts
                                        mov ax,cs
                                        mov ds,ax       ;get our data segment
                                        mov bx,offset int_trp
                                        call pmsg
                                        hlt             ;hardstop
                                ;!                                      !
                                ;---------------------------------------
                                        ENDIF   ;not loader_bios

                                ;*****************************************
                                ;*                                       *
                                ;*   CP/M Character I/O Interface Routines  *
                                ;*   Console is Usart (i8251a) on iSBC 86/12 *
                                ;*   at ports D8/DA                      *
                                ;*                                       *
                                ;*****************************************

                                CONST:          ;console status
126A E4DA                               in al,csts
126C 2402                               and al,2
126E 7402       1272                    jz const_ret
1270 0CFF                               or al,255       ;return non-zero if RDA
                                const_ret:
1272 C3                                 ret             ;Receiver Data Available

                                CONIN:                  ;console input
1273 E8F4FF     126A                    call const
1276 74FB       1273                    jz CONIN        ;wait for RDA
1278 E4D8                               in al,cdata
127A 247F                               and al,7fh      ;read data and remove parity bit
127C C3                                 ret

                                CONOUT:         ;console output
127D E4DA                               in al,csts
127F 2401                               and al,1        ;get console status
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B X 579
PACIFIC GROVE, CA 93950
SER. #____

```
                                
1281 74FA       127D            jz CONOUT       ;wait for TBE
1283 8AC1                       mov al,cl
1285 E6D8                       out cdata,al    ;Transmitter Buffer Empty
1287 C3                         ret             ;then return data
                                
                        LISTOUT:                ;list device output
                                
                                IF      blc_list
                        ;---------------------------------------
                        ;:                                     :
1288 E80700     1292            call LISTST
128B 74FB       1288            jz LISTOUT      ;wait for printer not busy
128D 8AC1                       mov al,cl
128F E640                       out ldata,al    ;send char to TI 810
                        ;:                                     :
                        ;---------------------------------------
                                ENDIF   ;blc_list
                                
1291 C3                         ret
                                
                        LISTST:                 ;poll list status
                                
                                IF      blc_list
                        ;---------------------------------------
                        ;:                                     :
1292 E441                       in al,lsts
1294 2481                       and al,81h      ;look at both TxRDY and DTR
1296 3C81                       cmp al,81h
1298 750A       12A4            jnz zero_ret    ;either false, printer is busy
129A 0CFF                       or al,255       ;both true, LPT is ready
                        ;:                                     :
                        ;---------------------------------------
                                ENDIF   ;blc_list
                                
129C C3                         ret
                                
                        PUNCH:  ;not implemented in this configuration
                        READER:
129D B01A                       mov al,1ah
129F C3                         ret             ;return EOF for now
                                
                        GETIOBF:
12A0 B000                       mov al,0        ;TTY: for consistency
12A2 C3                         ret             ;IOBYTE not implemented
                                
                        SETIOBF:
12A3 C3                         ret             ;iobyte not implemented
                                
                        zero_ret:
12A4 2400                       and al,0
12A6 C3                 ret                     ;return zero in AL and flags
                                
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        ; Routine to get and echo a console character
                        ;       and shift it to upper case
                        
                        uconecho:
12A7 E8C9FF     1273            call CONIN      ;get a console character
12AA 50                         push ax
12AB 8AC8                       mov cl,al       ;save and
12AD E8CDFF     127D            call CONOUT
12B0 58                         pop ax          ;echo to console
12B1 3C61                       cmp al,'a'
12B3 7206       12BB            jb uret         ;less than 'a' is ok
12B5 3C7A                       cmp al,'z'
12B7 7702       12BB            ja uret         ;greater than 'z' is ok
12B9 2C20                       sub al,'a'-'A'  ;else shift to caps
                        uret:
12BB C3                         ret
                        
                        ;       utility subroutine to print messages
                        
                        pmsg:
12BC 8A07                       mov al,[BX]     ;get next char from message
12BE 84C0                       test al,al
12C0 7428       12EA            jz return       ;if zero return
12C2 8AC8                       mov CL,AL
12C4 E8B6FF     127D            call CONOUT     ;print it
12C7 43                         inc BX
12C8 EBF2       12BC            jmps pmsg       ;next character and loop
                        
                        ;***********************************************
                        ;*                                             *
                        ;*          Disk Input/Output Routines         *
                        ;*                                             *
                        ;***********************************************
                        
                        SELDSK:         ;select disk given by register CL
12CA BB0000                     mov bx,0000h
12CD 80F902                     cmp cl,2        ;this BIOS only supports 2 disks
12D0 7318       12EA            jnb return      ;return w/ 0000 in BX if bad drive
12D2 B080                       mov al, 80h
12D4 80F900                     cmp cl,0
12D7 7502       12DB            jne sel1        ;drive 1 if not zero
12D9 B040                       mov al, 40h     ;else drive is 0
12DB A22E15             sel1:   mov sel_mask,al ;save drive select mask
                                                ;now, we need disk parameter address
12DE B500                       mov ch,0
12E0 8BD9                       mov bx,cx       ;BX = word(CL)
12E2 B104                       mov cl,4
12E4 D3E3                       shl bx,cl       ;multiply drive code * 16
                                ;create offset from Disk Parameter Base
12E6 81C34115                   add bx,offset dp_base
                        return:
12EA C3                         ret
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        HOME:   ;move selected disk to home position (Track 0)
12EB C606311500                 mov trk,0       ;set disk i/o to track zero
12F0 BB3315                     mov bx,offset hom_com
12F3 E83500     132B            call execute
12F6 74F2       12EA            jz return       ;home drive and return if OK
12F8 BB2F14                     mov bx,offset bad_hom   ;else print
12FB E8BEFF     12BC            call pmsg       ;"Home Error"
12FE EBEB       12EB            jmps home       ;and retry

                        SETTRK: ;set track address given by CX
1300 890E3115                   mov trk,cl      ;we only use 8 bits of track address
1304 C3                         ret

                        SETSEC: ;set sector number given by cx
1305 890E3215                   mov sect,cl     ;we only use 8 bits of sector address
1309 C3                         ret

                        SECTRAN: ;translate sector CX using table at [DX]
130A 8BD9                       mov bx,cx
130C 03DA                       add bx,dx       ;add sector to tran table address
130E 8A1F                       mov bl,[bx]     ;get logical sector
1310 C3                         ret

                        SETDMA: ;set DMA offset given by CX
1311 890E2A15                   mov dma_adr,CX
1315 C3                         ret

                        SETDMAB: ;set DMA segment given by CX
1316 890E2C15                   mov dma_seg,CX
131A C3                         ret
                        ;
                        GETSEGT:  ;return address of physical memory table
131B BB3815                     mov bx,offset seg_table
131E C3                         ret

                        ;***********************************************
                        ;*                                             *
                        ;*  All disk I/O parameters are setup:  the    *
                        ;*  Read and Write entry points transfer one   *
                        ;*  sector of 128 bytes to/from the current    *
                        ;*  DMA address using the current disk drive   *
                        ;*                                             *
                        ;***********************************************

                        READ:
131F B012                       mov al,12h      ;basic read sector command
1321 EB02       1325            jmps r_w_common

                        WRITE:
1323 B00A                       mov al,0ah      ;basic write sector command
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        r_w_common:
1325 8B2F15                     mov bx,offset io_com ;point to command string
1328 884701                     mov byte ptr 1[BX],al ;put command into string
                        ;       fall into execute and return

                        execute:  ;execute command string,
                                ;[BX] points to length,
                                ;       followed by Command byte,
                                ;       followed b  length-1 parameter bytes

132B 891E2815                   mov last_com,BX ;save command address for retries
                        outer_retry:
                                ;allow some retrying
132F C60627150A                 mov rtry_cnt,max_retries
                        retry:
1334 8B1E2815                   mov BX,last_com
1338 E88900     13C4            call send_com   ;transmit command to i8271
                        ;       check status poll

133B 8B1E2815                   mov BX,last_com
133F 8A4701                     mov al,1[bx]    ;get command op code
1342 B90008                     mov cx,0800h    ;mask if it will be 'int req'
1345 3C2C                       cmp al,2ch
1347 720B       1354            jb exec_poll    ;ok if it is an interrupt type
1349 B78080                     mov cx,8080h    ;else we use 'not command busy'
134C 240F                       and al,0fh
134E 3C0C                       cmp al,0ch      ;unless there isn't
1350 B000                       mov al,0
1352 7736       138A            ja exec_exit    ;       any result
                                                ;poll for bits in CH,
                        exec_poll:              ;  toggled with bits in CL

1354 E4A0                       in al,fdc_stat  ;read status
1356 22C5                       and al,ch
1358 32C1                       xor al,cl       ;  isolate what we want to poll
135A 74F8       1354            jz exec_poll    ;and loop until it is done

                                                ;Operation complete,
135C E4A1                       in al,fdc_rslt  ; see if result code indicates error
135E 241E                       and al,1eh
1360 7428       138A            jz exec_exit    ;no error, then exit
                                                ;some type of error occurred . . .
1362 3C10                       cmp al,10h
1364 7425       138B            je dr_nrdy      ;was it a not ready drive ?
                                                ;no,
                        dr_rdy: ; then we just retry read or write
1366 FE0E2715                   dec rtry_cnt
136A 75C8       1334            jnz retry       ;  up to 10 times

                        ;       retries do not recover from the
                        ;       hard error

```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #____

```
                                mov ah,0
                                mov bx,ax       ;make error code 16 bits
                                mov bx,errtbl[BX]
                                call pmsg       ;print appropriate message
                                in al,cdata     ;flush usart receiver buffer
                                call uconecho   ;read upper case console character
                                cmp al,'C'
                                je wboot_1      ;cancel
                                cmp al,'R'
                                je outer_retry  ;retry 10 more times
                                cmp al,'I'
                                je z_ret        ;ignore error
                                or al,255       ;set code for permanent error
                            exec_exit:
                                ret

                            dr_nrdy:        ;here to wait for drive ready
                                call test_ready
                                jnz retry       ;if it's ready now we are done
                                call test_ready
                                jnz retry       ;if not ready twice in row,
                                mov bx,offset nrdymsg
                                call pmsg ;"Drive Not Ready"
                            nrdy01:
                                call test_ready
                                jz nrdy01       ;now loop until drive ready
                                jmps retry      ;then go retry without decrement
                            zret:
                                and al,0
                                ret             ;return with no error code

                            wboot_1:            ;can't make it w/ a short leap
                                jmp WBOOT

                            ;***********************************************
                            ;*                                             *
                            ;*  The i8271 requires a read status command   *
                            ;*  to reset a drive-not-ready after the       *
                            ;*  drive becomes ready                        *
                            ;*                                             *
                            ;***********************************************

                            test_ready:
                                mov dh, 40h     ;proper mask if dr 1
                                test sel_mask,80h
                                jnz nrdy2
                                mov dh, 04h     ;mask for dr 0 status bit
                            nrdy2:
                                mov bx,offset rds_com
                                call send_com
                            dr_poll:
                                in al,fdc_stat  ;get status word
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                test al,80h
13BB A880                       jnz dr_poll     ;wait for not command busy
13BD 75FA       13B9            in al,fdc_rslt  ;get "special result"
13BF E4A1                       test al,dh      ;look at bit for this drive
13C1 84C6                       ret             ;return status of ready
13C3 C3

                        ;***********************************************
                        ;*                                             *
                        ;*  Send_com sends a command and parameters    *
                        ;*  to the i8271:  BX addresses parameters.    *
                        ;*  The DMA controller is also initialized     *
                        ;*  if this is a read or write                 *
                        ;*                                             *
                        ;***********************************************

                        send_com:
13C4 E4A0                       in al,fdc_stat
13C6 A880                       test al,80h     ;insure command not busy
13C8 75FA       13C4            jnz send_com    ;loop until ready

                                ;see if we have to initialize for a DMA operation

13CA 8A4701                     mov al,1[bx]    ;get command byte
13CD 3C12                       cmp al,12h
13CF 7504       13D5            jne write_maybe ;if not a read it could be write
13D1 B140                       mov cl,40h
13D3 EB06       13DB            jmps init_dma   ;is a read command, go set DMA
                        write_maybe:
13D5 3C0A                       cmp al,0ah
13D7 7520       13F9            jne dma_exit    ;leave DMA alone if not read or write
13D9 B180                       mov cl,80h      ;we have write, not read
                        init_dma:
                        ;we have a read or write operation, setup DMA controller
                        ;       (CL contains proper direction bit)
13DB B004                       mov al,04h
13DD E6A8                       out dmac_mode,al    ;enable dmac
13DF B000                       mov al,00
13E1 E6A5                       out dmac_cont,al    ;send first byte to control port
13E3 8AC1                       mov al,cl
13E5 E6A5                       out dmac_cont,al    ;load direction register
13E7 A12A15                     mov ax,dma_adr
13EA E6A4                       out dmac_adr,al     ;send low byte of DMA
13EC 8AC4                       mov al,ah
13EE E6A4                       out dmac_adr,al     ;send high byte
13F0 A12C15                     mov ax,dma_seg
13F3 E6AA                       out fdc_segment,al ;send low byte of segment address
13F5 8AC4                       mov al,ah
13F7 E6AA                       out fdc_segment,al ;then high segment address
                        dma_exit:
13F9 8A0F                       mov cl,[BX]     ;get count
13FB 43                         inc BX
13FC 8A07                       mov al,[BX]     ;get command
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                        or al,sel_mask  ;merge command and drive code
1402 E6A0                               out fdc_com,al  ;send command byte
                                parm_loop:
1404 FEC9                               dec cl
1406 7482               138A            jz exec_exit    ;no (more) parameters, return
1408 43                                 inc BX          ;point to (next) parameter
                                parm_poll:
1409 E4A0                               in al,fdc_stat
140B A820                               test al,20h     ;test "parameter register full" bit
140D 75FA               1409            jnz parm_poll   ;idle until parm reg not full
140F 8A07                               mov al,[BX]
1411 E6A1                               out fdc_parm,al ;send next parameter
1413 EBEF               1404            jmps parm_loop  ;go see if there are more parameters

                                ;***********************************************
                                ;*                                             *
                                ;*              Data Areas                     *
                                ;*                                             *
                                ;***********************************************
  1415                          data_offset     equ offset $

                                        dseg
                                        org     data_offset     ;contiguous with code segment

                                        IF      loader_bios
                                ;---------------------------------------------
                                ;|                                           |
1415 0D0A0D0A                   signon  db      cr,lf,cr,lf
1419 43502F4D2D38               db      'CP/M-86 Version 1.1',cr,lf,0
     362056657273
     696F6E20312E
     310D0A00
                                ;|                                           |
                                ;---------------------------------------------
                                        ENDIF   ;loader_bios

                                        IF      not loader_bios
                                ;---------------------------------------------
                                ;|                                           |
                                signon  db      cr,lf,cr,lf
                                db      '  System Generated  - 11 Jan 81',cr,lf,0
                                ;|                                           |
                                ;---------------------------------------------
                                        ENDIF   ;not loader_bios

142F 0D0A486F6D65               bad_hom db      cr,lf,'Home Error',cr,lf,0
     204572726F72
     0D0A00
143E 0D0A496E7465               int_trp db      cr,lf,'Interrupt Trap Halt',cr,lf,0
     727275707420
     547261702048
     616C740D0A00
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                
                
1456 761476147614       errtbl  dw er0,er1,er2,er3
     7614
145E 86149614A314               dw er4,er5,er6,er7
     B414
1466 C714DB14ED14               dw er8,er9,erA,erB
     0215
146E 121576147614               dw erC,erD,erE,erF
     7614
                
1476 0D0A4E756C6C       er0     db  cr,lf,'Null Error ??',0
     204572726F72
     203F3F00
  1476                  er1     equ er0
  1476                  er2     equ er0
  1476                  er3     equ er0
1486 0D0A436C6F63       er4     db  cr,lf,'Clock Error :',0
     6B204572726F
     72203A00
1496 0D0A4C617465       er5     db  cr,lf,'Late DMA :',0
     20444D41203A
     00
14A3 0D0A49442043       er6     db  cr,lf,'ID CRC Error :',0
     524320457272
     6F72203A00
14B4 0D0A44617461       er7     db  cr,lf,'Data CRC Error :',0
     204352432045
     72726F72203A
     00
14C7 0D0A44726976       er8     db  cr,lf,'Drive Not Ready :',0
     65204E6F7420
     526561647920
     3A00
14DB 0D0A57726974       er9     db  cr,lf,'Write Protect :',0
     652050726F74
     656374203A00
14ED 0D0A54726B20       erA     db  cr,lf,'Trk 00 Not Found :',0
     3030204E6F74
     20466F756E64
     203A00
1502 0D0A57726974       erB     db  cr,lf,'Write Fault :',0
     65204661756C
     74203A00
1512 0D0A53656374       erC     db  cr,lf,'Sector Not Found :',0
     6F72204E6F74
     20466F756E64
     203A00
  1476                  erD     equ er0
  1476                  erE     equ er0
  1476                  erF     equ er0
  14C7                  nrdymsg equ er8
                
```

COPYRIGHT © 1981, 1982
Digital Research
P.O. Box 579
Pacific Grove, CA 93950
SER. #: ____

```
690
691  1527 00            rtry_cnt db 0   ;disk error retry counter
692  1528 0000          last_com dw 0   ;address of last command string
693  152A 0000          dma_adr  dw 0   ;dma offset stored here
694  152C 0000          dma_seg  dw 0   ;dma segment stored here
695  152E 40            sel_mask db 40h ;select mask, 40h or 80h
696
697                     ;       Various command strings for i8271
698
699  1530 03            io_com   db 3   ;length
700  1531 00            rd_wr    db 0   ;read/write function code
701  1532 00            trk      db 0   ;track #
702  1533 00            sect     db 0   ;sector #
703
704  1534 022900        hom_com db 2,29h,0      ;home drive command
705  1537 012C          rds_com db 1,2ch        ;read status command
706
707                     ;       System Memory Segment Table
708
709  1539 02            segtable db 2   ;2 segments
710  153A AB01                   dw tpa_seg      ;1st seg starts after BIOS
711  153C 5506                   dw tpa_len      ;and extends to 08000
712  153E 0020                   dw 2000h        ;second is 20000 -
713  1540 0020                   dw 2000h        ;3FFFF (128K)
714
715  =                           include singles.lib ;read in disk definitions
716  =                  ;                DISKS 2
717  =  1541            dpbase  equ      $              ;Base of Disk Parameter Blocks
718  =1541 70150000     dpe0    dw       xlt0,0000h     ;Translate Table
719  =1545 00000000             dw       0000h,0000h    ;Scratch Area
720  =1549 8A156115             dw       dirbuf,dpb0    ;Dir Buff, Parm Block
721  =154D 29160A16             dw       csv0,alv0      ;Check, Alloc Vectors
722  =1551 70150000     dpe1    dw       xlt1,0000h     ;Translate Table
723  =1555 00000000             dw       0000h,0000h    ;Scratch Area
724  =1559 8A156115             dw       dirbuf,dpb1    ;Dir Buff, Parm Block
725  =155D 58163916             dw       csv1,alv1      ;Check, Alloc Vectors
726  =                  ;                DISKDEF 0,1,26,6,1024,243,64,64,2
727  =                  ;
728  =                  ;        1944: 128 Byte Record Capacity
729  =                  ;         243: Kilobyte Drive  Capacity
730  =                  ;          64: 32 Byte Directory Entries
731  =                  ;          64: Checked Directory Entries
732  =                  ;         128: Records / Extent
733  =                  ;           8: Records / Block
734  =                  ;          26: Sectors / Track
735  =                  ;           2: Reserved  Tracks
736  =                  ;           6: Sector Skew Factor
737  =                  ;
738  =  1561            dpb0    equ      offset $       ;Disk Parameter Block
739  =1561 1A00                 dw       26             ;Sectors Per Track
740  =1563 03                   db       3              ;Block Shift
741  =1564 07                   db       7              ;Block Mask
742  =1565 00                   db       0              ;Extnt Mask
```

COPYRIGHT 1981 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                dw      242             ;Disk Size - 1
                                dw      63              ;Directory Max
                                db      192             ;Alloc0
                                db      0               ;Alloc1
                                dw      16              ;Check Size
                                dw      2               ;Offset
                        xlt0    equ     offset $        ;Translate Table
                                db      1,7,13,19
                                db      25,5,11,17
                                db      23,3,9,15
                                db      21,2,8,14
                                db      20,26,6,12
                                db      18,24,4,10
                                db      16,22
                        cks0    equ     31              ;Allocation Vector Size
                        css0    equ     16              ;Check Vector Size
                        ;               DISKDEF 1,0
                        ;
                        ;       Disk 1 is the same as Disk 0
                        ;
                        dpb1    equ     dpb0            ;Equivalent Parameters
                        als1    equ     als0            ;Same Allocation Vector Size
                        css1    equ     css0            ;Same Checksum Vector Size
                        xlt1    equ     xlt0            ;Same Translate Table
                        ;               ENDEF
                        ;
                        ;       Uninitialized Scratch Memory Follows:
                        ;
                        begdat  equ     offset $        ;Start of Scratch Area
                        dirbuf  rs      128             ;Directory Buffer
                        alv0    rs      als0            ;Alloc Vector
                        csv0    rs      css0            ;Check Vector
                        alv1    rs      als1            ;Alloc Vector
                        csv1    rs      css1            ;Check Vector
                        enddat  equ     offset $        ;End of Scratch Area
                        datsiz  equ     offset $-begdat ;Size of Scratch Area
                                db      0               ;Marks End of Module


                        loc_stk rw  32  ;local stack for initialization
                        stkbase equ offset $

                        lastoff equ offset $
                        tpa_seg equ (lastoff+0400h+15) / 16
                        tpa_len equ 0800h - tpa_seg
                                db 0    ;fill last address for GENCMD

                        ;***********************************************
                        ;*                                             *
                        ;*         Dummy Data Section                  *
                        ;*                                             *
                        ;***********************************************
```

COPYRIGHT © 1981, 1982
Digital Research
P.O. Box 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                dseg    0       ;absolute low memory
                                org     0       ;(interrupt vectors)
0000                    int0_offset     rw      1
0002                    int0_segment    rw      1
                        ;       pad to system call vector
0004                            rw      2*(bdos_int-1)

0380                    bdos_offset     rw      1
0382                    bdos_segment    rw      1
                                END
```

END OF ASSEMBLY. NUMBER OF ERRORS: 0. USE FACTOR: 11%

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
ALS0            001F N   758#  765   774
ALS1            001F N   765#  776
ALV0            160A V   721   774#
ALV1            1639 V   725   773#
BADHOM          142F V   379   630#
BASE204         00A0 N    87#   89    90    91    92    93    94    95    96    97
                          98    99   100   101   102
BDOSINT         00E0 N    45#  802
BDOSOFFSET      0380 V   179   211   804#
BDOSOFST        0406 N    52#   63#  179   211
BDOSSEGMENT     0382 V   212   805#
BEGDAT          158A N   772#  779
BIOSCODE        1200 N    50#   61#  113
BLCLIST         FFFF N    44#   71   274   289
BLCRESET        0060 N    76#  192
CCP             0003 L   112#  222   224
CCPOFFSET       0003 N    51#   62#  111
CDATA           00D8 N    69#  259   269   433
CONIN           1273 L   124   256#  258   324
CONOUT          127D L   125   263#  267   327   344
CONST           126A L   123   248#  257
CONSTRET        1272 L   251   253#
CR              000D N   106#  612   612   613   624   624   625   630   630   633
                         633   648   654   657   660   663   667   671   674   678
                         681
CSS0            0010 N   759#  766   775
CSS1            0010 N   766#  777
CSTS            00DA N    68#  249   264
CSV0            1629 V   721   775#
CSV1            1658 V   725   777#
DATAOFFSET      1415 N   604#  607
DATSIZ          00DE N   777#
DIRBUF          158A V   720   724   773#
DMAADR          152A V   398   572   693#
DMACADR         00A4 N    94#  573   575
DMACCONT        00A5 N    95#  569   571
DMACMODE        00A8 N    98#  567
DMACSADR        00A7 N    97#
DMACSCAN        00A6 N    96#
DMACSTAT        00A8 N    99#
DMAEXIT         13F9 L   561   580#
DMASEG          152C V   402   576   694#
DPB0            1561 N   720   738#  764
DPB1            1561 N   724   764#
DPBASE          1541 L   369   717#
DPE0            1541 V   718#
DPE1            1551 V   722#
DRNRDY          138B L   469   475#
DRPOLL          13B9 L   529#  533
DRRDY           1366 L   471#
ENDDAT          1668 N   778#
ER0             1476 V   639   648#  651   652   653   685   686   687
ER1             1476 V   639   651#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
ER2              1476 V   639   652#
ER3              1476 V   639   653#
ER4              1486 V   641   654#
ER5              1496 V   641   657#
ER6              14A3 V   641   660#
ER7              14B4 V   641   663#
ER8              14C7 V   643   667#  688
ER9              14DB V   643   671#
ERA              14ED V   643   674#
ERB              1502 V   643   678#
ERC              1512 V   645   681#
ERD              1476 V   645   685#
ERE              1476 V   645   686#
ERF              1476 V   645   687#
ERRTBL           1456 V   481   639#
EXECEXIT         138A L   454   466   492#  589
EXECPOLL         1354 L   449   456#  461
EXECUTE          132B L   377   431#
FALSE            0000 N    30#
FDCCOM           00A0 N    89#  536
FDCPARM          00A1 N    91#  596
FDCRSLT          00A1 N    92#  464   534
FDCRST           00A2 N    93#
FDCSEGMENT       00AA N   101#  577   579
FDCSEL           00A9 N   100#
FDCSTAT          00A0 N    90#  458   530   548   592
GETIOBF          12A0 L   140   308#
GETSEGT          131B L   139   405#
HOMCOM           1533 V   376   704#
HOME             12EB L   129   374#  381
INIT             123F L   121   150#
INITDMA          13DB L   558   563#
INTOOFFSET       0000 V   172   799#
INTOSEGMENT      0002 V   173   800#
INTTRP           143E V   233   633#
IOCOM            152F V   427   699#
LASTCOM          1528 V   436   441   445   692#
LASTOFF          16A9 N   786#  787
LDATA            0040 N    75#  194   196   198   280
LF               000A N   108#  612   612   613   624   624   625   630   630   633
                          633   648   654   657   660   663   667   671   674   678
                          681
LISTOUT          1288 L   126   272#  278
LISTST           1292 L   136   277   287#
LOADERBIOS       FFFF N    43#   47    58   159   203   226   609   621
LOCSTK           1669 V   783#
LSTS             0041 N    74#  292
MAXRETRIES       000A N   104#  439
NRDY01           139B L   502#  504
NRDY2            13B3 L   524   526#
NRDYMSG          14C7 V   500   688#
OUTERRETRY       132F L   437#  488
PARMLOOP         1404 L   587#  597
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
PARM'OLL        1409 L  591#  594
PMSG            12BC L  220   234   339#  346   380   482   501
PUNCH           129D L  127   303#
RDSCOM          1536 V  527   705#
RDWR            1530 V  700#
READ            131F L  134   418#
READER          129D L  128   304#
RESET204        00AF N  102#
RETRY           1334 L  440#  473   497   499   505
RETURN          12EA L  342   357   370#  378
RTRYCNT         1527 V  439   472   691#
RWCOMMON        1325 L  420   426#
SECT            1532 V  388   702#
SECTRAN         130A L  137   391#
SEGTABLE        1538 V  406   709#
SEL1            12DB L  360   362#
SELDSK          12CA L  130   354#
SELMASK         152E V  362   523   585   695#
SENDCOM         13C4 L  442   528   547#  550
SETDMA          1311 L  133   397#
SETDMAB         1316 L  138   401#
SETIOBF         12A3 L  141   312#
SETSEC          1305 L  132   387#
SETTRK          1300 L  131   383#
SIGNON          1415 V  219   612#  624#
STKBASE         16A9 N  156   784#
TESTREADY       13A8 L  496   498   503   521#
TPALEN          0655 N  711   783#
TPASEG          01AB N  710   787#  788
TRK             1531 V  375   384   701#
TRUE            FFFF N   29#   30    43    44
UCONECHO        12A7 L  323#  484
URET            12BB L  330   332   334#
WBOOT           1267 L  122   224#  511
WBOOTL          13A5 L  486   510#
WRITE           1323 L  135   422#
WRITEMAYBE      13D5 L  556   559#
XLT0            1570 N  718   750#  767
XLT1            1570 N  722   767#
ZERORET         12A4 L  295   315#
ZRET            13A2 L  490   506#
```

cp/m 86 v.1.1 - vol I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
U.S.A.
PACIFIC GROVE, CA 93950
SER. # C81-162